Міністерство освіти і науки України

Луцький національний технічний університет
(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій
(повне найменування факультету)

Кафедра комп'ютерної інженерії та кібербезпеки
(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

ДОДАТОК ДОПОВНЕНОЇ РЕАЛЬНОСТІ В СЕРЕДОВИЩІ UNITY3D ЗАСОБАМИ .NET

AN AUGMENTED REALITY APPLICATION IN THE UNITY3D ENVIRONMENT USING .NET TOOLS

спеціальність 123 Комп'ютерна інженерія
(шифр і назва спеціальності)

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІ-41
Панько Максим Іванович

(підпис)

Керівник:
к.т.н., доцент
Мельник Катерина Вікторівна

(підпис)

Кваліфікаційну роботу
допущено до захисту
« 12 » червня 2024 р.
Гарант освітньої програми:
к.т.н., доцент
Лавренчук Світлана Василівна

(підпис)

Луцьк – 2024 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерної інженерії та кібербезпеки*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *123 Комп'ютерна інженерія*
Освітня програма: *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ проф. Н.Чернящук
« 10 » 01 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Паньку Максиму Івановичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Додаток доповненої реальності в середовищі Unity3D засобами .NET*

Керівник роботи *к.т.н., доцент Мельник Катерина Вікторівна*

затверджені наказом закладу вищої освіти від «30» грудня 2023 року № 459/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 11.06.2024р.

3. Вихідні дані до роботи *Джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання, опубліковані зарубіжні та вітчизняні роботи в даній області та різні інтернет-ресурси технічного спрямування*

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

Аналіз предметної області технології доповненої реальності , огляд та вибір існуючих засобів реалізації AR, розробка додатку, оцінка результатів дослідження

5. Перелік графічного (ілюстративного) матеріалу:

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз проблеми за темою роботи та постановка завдань дослідження	*Мельник К.В., доцент*		
Теоретичне дослідження та практична реалізація	*Мельник К.В., доцент*		
Практична реалізація AR-додатку	*Мельник К.В., доцент*		
Нормоконтроль	*Багнюк Н.В., доцент*		
Гарант ОП	*Лавренчук С.В., доцент*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	*Міскевич О.І.,асистент*		

7. Дата видачі завдання 10.01.2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Провести огляд літератури із досліджуваної проблеми*	до 17.02.2024 р.	Виконано
2.	*Провести аналіз інструментів та засобів розробки AR-застосунків*	до 20.03.2024 р.	Виконано
3.	*Розробка AR-додатку в середовищі Unity засобами Vuforia SDK*	до 10.05.2024 р.	Виконано
4.	*Висновки та пропозиції*	до 12.05.2025 р.	Виконано
5.	*Формування списку використаних джерел*	до 15.05.2024 р.	Виконано
6.	*Формування додатків*	до 17.05.2024 р.	Виконано
7.	*Оформлення ілюстративного матеріалу*	до 23.05.2024 р.	Виконано
8.	*Нормоконтроль*	до 01.06.2024 р.	Виконано
9.	*Інструментальна перевірка на академічний плагіат*	до 04.06.2024 р	Виконано
10.	*Представлення кваліфікаційної роботи бакалавра до захисту*	до 11.06.2024 р.	Виконано

Здобувач вищої освіти _____ (підпис) Панько М.І. (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ (підпис) Мельник К.В. (прізвище, ініціали)

АНОТАЦІЯ

Панько М.І. Розробка додатку доповненої реальності в середовищі Unity3D засобами .NET.

Кваліфікаційна робота бакалавра ОП «Комп'ютерна інженерія» спеціальності 123 Комп'ютерна інженерія. Луцький національний технічний університет. Луцьк, 2024.

Кваліфікаційна робота складається з вступу, трьох розділів, висновків, списку використаних джерел, трьох додатків.

Дана робота присвячена розробці додатку доповненої реальності з метою оптимізації виробництва, шляхом інтегрування в реальне середовище віртуальних інструкцій та інформації.

У цій роботі будуть розглянути готові рішення, а також процес розробки програмної частини застосунку на базі рушія Unity, мовою програмування C# з використанням додаткових фреймворків для розпізнавання рухів рук – ManoMotion, та багатопотокової обробки даних – UniTask.

В результаті додаток доповнюватиме реальне середовище покроковими інструкціїями щодо збірки виробу, що дозволить виробникам скоротити кількість браку, та час на навчання нового фахівця.

Ключові слова: Доповнена реальність, оптимізація виробництва Unity, ManoMotion.

ANNOTATION

Panko M.S. Development of an augmented reality application in the Unity3D environment using .NET tools.

Qualifying work of a bachelor of EP "Computer Engineering" specialty 123 Computer Engineering. Lutsk National Technical University. Lutsk, 2024.

The qualification work consists of an introduction, three sections, conclusions, a list of used sources, and three appendices.

This work is devoted to the development of an augmented reality application for the purpose of optimizing production by projecting instructions and information on assembly in a real environment.

This work will consider ready-made solutions, as well as the process of developing the software part of the application based on the Unity engine, in the C# programming language, using additional frameworks for recognizing hand movements - ManoMotion, and multi-threaded data processing - UniTask.

As a result, the application will supplement the real environment with step-by-step instructions for assembling the product, which will allow manufacturers to reduce the amount of defects and time for training a new specialist.

Keywords: Augmented reality, Unity production optimization, Manomotion.

ЗМІСТ

ВСТУП

Актуальність теми. Технології доповненої реальності розвиваються стрімкими темпами набувачи все більшої популярності [1], що відкриває нові перспективи для їх використання у різноманітних сферах. Впровадження таких технологій вже сьогодні показує вражаючі результати.

Як свідчить досвід компанії, Lockheed Martin використання технологій доповненої реальності значною мірою впливає на оптимізацію виробничих процесів. За даними їхніх досліджень [2], впровадження AR технологій дозволило зменшити час вивчення інструкцій зі складання на 95%, крім того, скоротити загальний час навчання на 85% та збільшити продуктивність більш ніж на 40%. Що є важливим показником для будь-якої компанії, оскільки це безпосередньо впливає на її конкурентоспроможність і прибутковість. Отже, впровадження технологій доповненої реальності в сучасний виробничий процес є надзвичайно ефективним та необхідним кроком. Якому слідують все більше виробників, модернізуючи внутрішні процеси відповідно до принципів четвертої промислової революції [3].

Метою роботи є розробка прототипу додатку для покращення виробничих процесів, шляхом проєктування інструкцій та інформації щодо збірки продукту в реальне середовище, за допомогою 3D графіки.

Об’єкт дослідження – процес та результат розробки кросплатформеного AR застосунку на платформі Unity з використанням інтуїтівного інтерфейсу користувача, реалізованого за допомогою машинного зору.

Предмет дослідження – використання технології доповненої реальності для оптимізації виробництва.

Завдання, які необхідно виконати:

– здійснити огляд доступних інструментів для пришвидшення розробки, обрати найоптимальніші;

– реалізувати основні компоненти додатку: сканер QR-кодів, жестовий інтерфейс користувача для взаємодії з цифровими об’єктами на базі фреймворку ManoMotion та маркерне орієнтування віртуальної інформації в просторі;

– провести тестування та налагодження застосунку.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ AR ТЕХНОЛОГІЇ

1.1 Концепція доповненої реальності

Останнім часом технології доповненої реальності стають одним з найперспективніших напрямків IT-розробок [4]. Ця технологія представляє собою новий спосіб подання та отримання інформації людиною, що суттєво спрощує її сприйняття.

Доповнена реальність (argumented reality) – це технологія, що інтегрує цифрову інформацію у фізичне середовище користувача. Основна ідея полягає у трансляції в реальний світ цифрових даних у режимі реального часу. Дані можуть бути представлені такими віртуальними об’єктами, як: текст, відео, 3D моделі, аудіо та тактильні відчуття. AR забезпечує миттєвий доступ до інформації, якої не існує в реальному середовищі, дозволяючи користувачеві взаємодіяти з комп’ютерним інтерфейсом безпосередньо навколо нього. Це дозволяє відмовитися від традиційних методів взаємодії, наприклад настільних моніторів, на користь інтерфейсів орієнтованих на реальне середовище. Таким чином, доповнена реальність перетворює навколишній світ на користувацький інтерфейс, як зображено на рисунку 1.1.

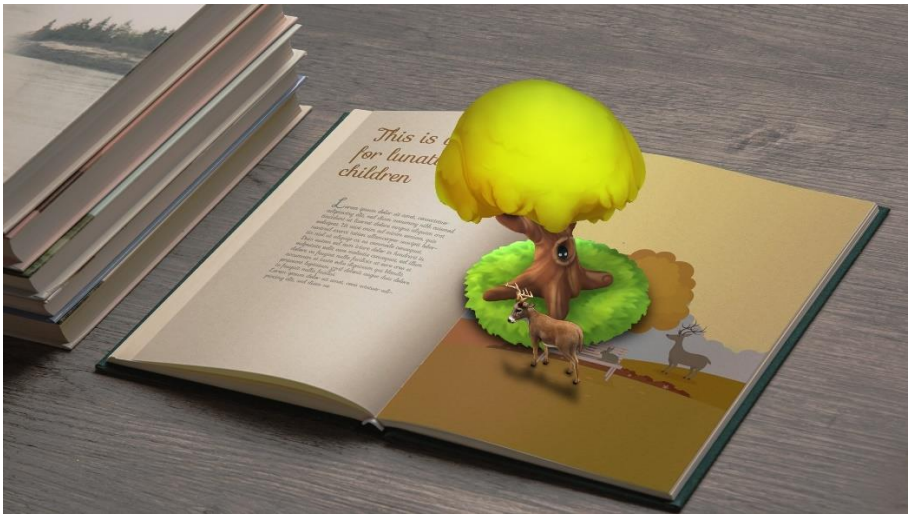


Рисунок 1.1 – Приклад роботи AR [5]

Щоб додати цифрові дані у реальне середовище, технологія AR потребує пристрій введення. Таким пристроєм може бути смартфон, планшет, комп'ютер з веб-камерою або окуляри доповненої реальності. Відеопотік з пристрою обробробляє рушій доповненої реальності в режимі реального часу, щоб в залежності від типу реалізаці інтегрувати віртуальну інформацію у реальне середовище.

1.2 Типи систем доповненої реальності

На даний час можна виділити два основних принципи роботи AR: marker-based AR – працюча на основі розпізнавання маркерів та maker-less AR – без маркерна доповнена реальність.

Маркерна AR (Marker-based AR) – це тип доповненої реальності, в якій вміст віртуального контенту залежить від виду ідентифікованого маркеру. Маркером називають об'єкт в навколишньому середовищі, що має заздалегідь передбачений вигляд [6]. Рушій розпізнає мітку та в залежності від отриманої інформації, може точно спозиціонувати цифровий об'єкт на маркер, що створює ефект його фізичної присутності. Часто в ролі маркера використовують QR-код (Quick Responsecode), але можливі й інші варіанти, такі як штрих код, круговий маркер, зображення та об'єкти реального світу, очі та навіть обличчя людини. На рисунку 1.2 наведено приклади міток.

Рисунок 1.2 – Приклади маркерів

Машинний зір є надзвичайно важливою технологією у даному типі AR, оскільки від його точності залежить якість функціонування системи, зокрема правильна інтерпретація маркера. Одним з популярних алгоритмів машинного зору в контексті розробки додатків доповненої реальності є feature detection (виявлення ознак). Цей метод на відміну від інших, найбільш стійкий до перешкод, трансформацій, що дозволяє знаходити об'єкти навіть при наявності фізичних перешкод. Feature detection спрямований на обчислення абстракцій зображення і виділення ключових особливостей. Не існує єдиного визначення ключової особливості зображення, різні алгоритми виділяти різні характеристики, до них можна віднести як ізольовані точки, так і схожі пов'язані області або криві.

Без маркерна AR (Marker-less AR) – це технологія, яка дозволяє накладати віртуальний вміст на реальний світ без попереднього знання середовища. Для роботи система використовує дані з різних датчиків пристрою, таких як компас, акселерометр, гіроскоп, GPS (Global Positioning System), щоб побудувати цифрову копію навколишнього простору та точно спозиціонувати віртуальний контент.

SLAM (Simultaneous Localization and Mapping) – технологія мобільного картографування, що дозволяє пристрою створювати карту навколишнього середовища, водночас визначаючи своє місцезнаходження на ній. Для обчислення відстані до об'єктів використовується показник глибини камери або технологія лазерного сканування LIDAR. LIDAR (Laser Imaging Detection And Ranging) – це метод визначення дальності за допомогою наведення променя лазера на поверхню об'єкта та вимірювання часу, протягом якого відбите світло повертається до приймача. По суті, кожен датчик, який можна використати для вимірювання фізичних величин, таких як місцезнаходження, положення в просторі, відстань або швидкість, відноситься до системи SLAM. На основі даних SLAM, marker-less AR розраховує просторове положення віртуального об'єкта.

Безмаркерна доповнена реальність має різні підвиди, серед яких можна виділити AR на основі накладання, проекції та розташування.

Геолокаційна AR (Location-based AR) – це тип доповненої реальності, в якій вміст або інформації пов'язана з конкретними GPS-координатами. Перебуваючи

у визначеному місці у користувача з’являється можливість взаємодіяти з віртуальним об’єктом. Це відкриває нові можливості для різних галузей, а саме: туризму, культури, маркетингу та розважальної сфери. Основною технічною проблемою використання location-based AR є похибка GPS. Існуючі навігаційні системи, дозволяють визначати місцезнаходження пристрою в реальному часі, однак навіть за ідеальних умов точність позиціонування коливається від 5 до 10 метрів. Погрішність впливає на синхронізацію положення віртуальної інформації та поля зору камери пристрою, що негавтино позначається на якості користувацького досвіду.

Проекційна AR (Projection-based AR) – це тип доповненої реальності, який використовує фізичні об’єкти, або площини для оптичного проектування зображення. Технологія дозволяє використовувати реальні об’єкти як дисплеї, на які накладається необхідна інформація. Взаємодія з проектованим контентом відбувається через фізичний контакт із поверхнею, на яку здійснюється проекція.

AR на основі накладання (Superimposition-based AR) – метод, який передбачає часткову або повну заміну вигляду оригінального об’єкта доповненим зображенням того ж об’єкта. У цьому типі AR важливу роль відіграє розпізнавання об’єктів, оскільки програма не може замінити оригінальний об’єкт на доповнений, якщо не вдається його ідентифікувати. Цей вид доповненої реальності був популяризований соціальними платформами, такими як Snapchat, Instagram, TikTok, які використовують його для створення масок та фільтрів.

1.3 Історія розвитку AR

Сам термін Augmented reality (доповнена реальність), ймовірно, вперше використали дослідники Boeing Aerospace Том Коделл (англ. Tom Caudell) і Девід Мізелл (англ. David Mizell) в 1990 році. Під час роботи над, наголовним дисплеєм, який додавав до поля зору оператора інформацію, пов’язану з завданням яке він виконує.

Хоча визначення доповненої реальності датуються 90-ми роками минулого століття, застосування та вивчення цих технологій почалося набагато раніше, перші згадки можна знайти вже наприкінці 1950-х років. Першим прикладом є симулятор Sensorama 1957 року, розроблений Мортоном Хейлінгом (англ. Morton Hayling). Цей симулятор, трохи більший за аркадний автомат, дозволяв користувачеві віртуально їздити вулицями Брукліна на мотоциклі. Посилюючи сприйняття такими відчуттями, як вібрація, вітер, зворотний зв'язок на кермі, стерео аудіо та спеціалізована система відтворення запахів.

У 1966 році Іван Сазерленд (англ. Ivan Sutherland) розробив перший шолом доповненої реальності, відомий як «Дамоклів меч» (рисунок 1.3). Пристрій був оснащений лінзами, які дозволяли переглядати зображення, накладені на реальний світ. Метою роботи була допомога пілотам гвинтокрилів у пілотуванні при умовах недостатньої видимості. Система відстежувала положення голови пілота, й передавала зображення з камер в оптично прозору гарнітуру. Свою назву «Дамоклів меч» механізм отримав, оскільки мав надмірну вагу, та для використання потребував кріплення до стелі за допомогою обертової руки.

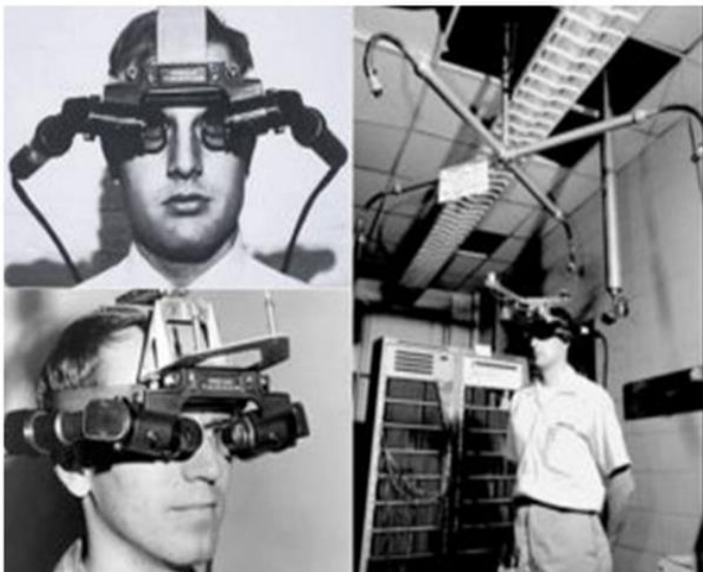


Рисунок 1.3 – AR шолом «Дамоклів меч» [7]

Пізніше у 1970-х роках Іван Сазерленд на базі університету Північної Кароліни реалізував GROPE, першу систему силового зворотного зв'язку, а Майрон Крюгер (англ. Myron Krueger) створив VIDEOPLACE – штучну реальність, яка дозволила користувачам спілкуватися у віртуальному середовищі, незважаючи на відстань. Система складалася із декількох темних кімнат із великими екранами. Де користувачі могли бачити свої силуети, що імітують їхні власні дії. Використовуючи камери, VIDEOPLACE відстежувала положення тіл користувачів та рухи, проектуючи їх на екран. Таким чином два або більше користувачів могли взаємодіяти у 2D-віртуальному просторі.

У 1982 році Томас Фернесс (англ. Thomas Furness) продемонстрував Visually Coupled Airbone System Simulator – симулятор візуально пов'язаних повітряно-десантних систем, більш відомий як шолом «Дарта Вейдера». Який був призначений для підготовки пілотів військово-повітряних сил США. Протягом 1986-1989 років Фернесс керував програмою Super Cockpit, яка базувалася на використанні природних механізмів сприйняття людини. За допомогою використання нашоломного дисплею була розроблена система, що проектувала 3D карти, перспективні інфрачервоні та радарні зображення, а також дані авіоніки в віртуальний простір, який пілот міг переглядати в реальному часі. Система відстеження положення шолома, голосові елементи керування та датчики дозволили пілоту керувати літаком жестами, рухами очей, що в свою чергу дозволило змеши складність і кількість елементів керування в кабіні.

Загалом, 1980-ті стали роками, коли почали з'являтися перші комерційні пристрої. У 1985 році компанія VPL випустила DataGlove, рукавички з датчиками які могли вимірювати згинання пальців, орієнтацію та положення, а також ідентифікувати жести руками. Наприкінці 80-х років Fake Space Labs створили бінокулярний всеорієнтаційний монітор BOOM, що складався з пристрою стереоскопічного відображення, що забезпечує рухоме віртуальне середовище. Завдяки BOOM і DataGlove дослідницький центр NASA Ames розробив віртуальну аеродинамічну трубу зображену на рисунку 1.4, щоб досліджувати

вплив аеродинамічного потоку на керування віртуальними літаками та космічними апаратами.

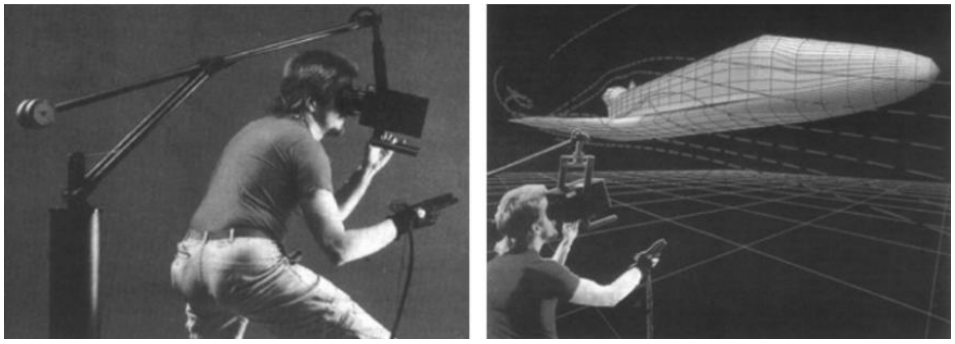


Рисунок 1.4 – Використання AR космічним агенством NASA [8]

У 1992 році Луї Розенберг (англ. Louis Rosenberg) розробив «Virtual Fixture», першу функціональну платформу доповненої реальності, яка дозволяла керувати рухом маніпулятора, переміщаючи руки. У 1993 році Джулі Мартін (англ. Julie Martin) проводить першу театральну постановку доповненої реальності – «Танці в кіберпросторі». Під час вистави акробати танцювали у віртуальних об’єктах на сцені та навколо них. Через кілька років у 1997 Стівен Файнер (англ. Steve Feiner) розробив портативну систему «Mobile AR MARS» зображену на рисунку 1.5, яка додавала віртуальну інформацію про туристичні будівлі.

Рисунок 1.5 – Портативна система «Mobile AR MARS» [9]

У 2000 році Хірокадзу Като (англ. Hirokazu Kato) створив набір програмних бібліотек з відкритим вихідним кодом – «ARToolKit». Для розробки додатків доповненої реальності.

У 2008 році був запущений перший мобільний додаток доповненої реальності – Wikitude World Browser. Wikitude надавав користувачам інформацію про пам’ятки та визначні місця на основі місцезнаходження, накладаючи її на зображення реального світу, яке видно через камеру пристрою.

1.4 Сфери застосування AR

Доповнена реальність дедалі більше інтегрується у повсякденне життя, поєднуючи цифрову інформацію з фізичним середовищем. З розвитком AR, змінюються різні аспекти щоденного побуту. Підтвердженням цього є мобільні додатки, які використовують мільйони користувачів по всьому світі.

Наприклад, соціальні мережі, такі як Snapchat, Instagram, TikTok, FaceTime, впровадили фільтри, та ефекти на основі доповненої реальності, що дозволяють користувачам додавати до своїх фотографій та відео різноманітні віртуальні елементи. Це значно збільшує залученість та розширює можливості для самовираження та творчості.

Одним з найвидатніших прикладів AR в сфері розваг є гра Pokemon GO, яка була запущена у 2016 році. Протягом першого тижня загальна кількість завантажень з Google Play досягла 10 мільйонів. Крім того, гра принесла розробникам 3,6 мільярда доларів станом на 2024 рік [10]. Гравці шукали та ловили віртуальних покемонів, які з’являлися в реальному світі, переглядаючи їх через екран смартфона. Цей інтерактивний досвід не лише сприяв фізичній активності, але й стимулював досліджувати оточуюче середовище.

Також доповнена реальність відіграє важливу роль у сфері маркетингу. Презентація продукту в 3D форматі формує позитивне враження про товар та гарантує зацікавленість потенційного покупця. Одним із найвідоміших застосувань AR у сфері продаж є додаток IKEA Place. Ця програма дозволяє

користувачам розставляти віртуальні меблі у своїй оселі, наочно візуалізуючи як нові меблі вписуються у їхній інтер'єр, що збільшує ймовірність покупки. Іншим прикладом є Warby Parker. Додаток використовує AR, щоб дозволити клієнтам приміряти окуляри, не виходячи з дому. Це підвищує зручність і зменшує потребу відвідувати звичайний магазин, щоб знайти ідеальні оправи. Окрім окулярів, доповнена реальність особливо корисна в індустрії макіяжу. Бренд Loreal пропонує варіанти макіяжу на основі AR, які дають змогу клієнтам віртуально нанести макіяж та підібрати правильний відтінок для свого тону шкіри. Це лише декілька найпопулярніших прикладів, вдалого використання AR в індустрії маркетингу.

AR в освітньому процесі покликана доповнити стандартну навчальну програму. Прикладом цього є різного роду симулятори та наочні моделі для навчання, які в реальному середовищі повязані з високим рівнем ризику або значними витратами. Додатково, AR використовується в підручниках з інтерактивним вмістом, які при скануванні їх за допомогою пристроїв надають додаткову інформацію, таку як 3D графіка, відео, текст або аудіо в режимі реального часу. Такий підхід не лише підвищує зацікавленість в навчанні, але й сприяє більш глибокому розумінню складних тем.

Інтеграція доповненої реальності в охорону здоров'я дозволяє, використовувати нові підходи до лікування пацієнтів. Одним з найважливіших внесків AR є її застосування в хірургії, де вона допомагає хірургам у плануванні та виконанні складних процедур. Накладаючи зображення з рентгенівських знімків або комп'ютерної томографії на тіло пацієнта, хірурги отримуть більш чітке уявлення про хірургічне поле що дозволяє знизити ризики, пов'язані з інвазивними втручаннями. Зокрема, технологія виявилася надзвичайно корисною в нейрохірургії, сприяючи безпечнішій та ефективнішій навігації складними структурами мозку.

РОЗДІЛ 2
ОГЛЯД ТА ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ AR

2.1 Вибір інтрументів для розробки AR-додатку

Ігровий рушій – це програмне забезпечення, платформа, створена для розробки комп'ютерних ігор, що надає розробникам необхідний функціонал для створення 2D або 3D-графіки, моделювання фізичної поведінки об'єктів, відтворення звуків, написання сценаріїв, анімацій, реалізації штучного інтелекту, організації мережевого зв'язку та інших складових, що формують ігровий досвід. Завдяки універсальності та можливості використання в різноманітних проектах, ігрові рушії відіграють значну роль у процесі створення ігор, дозволяючи розробникам зосередитися на створенні внутрішньої логіки та зовшнього вигляду проекту, використовуючи вже готові засоби та рішення.

Термін «ігровий рушій» з'явився в середині 1990-х років з виходом таких ігор, як «Doom» та «Quake» від однойменної компанії id Software [11]. Doom став настільки популярним, що це породило тенденцію до створювання ігор на його основі. Рушій дозволяв додавати нові рівні та модифікації, не змінюючи основний код гри, що заклало основу для сучасних технологій та визначило напрямок розвитку індустрії. У 1996 році компанія id Software випустила «Quake», першу гру з повністю тривимірною графікою, побудованою на полігональних моделях. Рушій Quake став основою для багатьох майбутніх ігор завдяки своїм передовим можливостям та модульності. Він був одним з перших рушіїв, який надав можливість ліцензованого використання іншим компаніям, що сприяло активному розвитку індустрії. Одночасно з цим з'явилися перші комерційні ігрові рушії. Наприклад, рушій «Unreal Engine», випущений Epic Games у 1998 році, став доступним для ліцензування іншими розробниками, що значно спростило процес створення нових ігор.

Незважаючи на специфіку назви, ігрові рушії використовуються не лише для розробки відеоігор. Вони також знайшли своє застосування в інших

інтерактивних додатках, які потребують додавання графіки в режимі реального часу. Поширеними ігровими рушіями, що мають широке застосування є:

Godot – це безкоштовна платформа з відкритим вихідним кодом, розроблена Массачусетським технологічним інститутом [12]. Рушій є кросплатформеним, що забезпечує можливість створювати ігри для Windows, Linux, MacOS, Android, iOS, веб-браузерів і консолей. Підтримуються як 32, так і 64-розрядні версії Windows, Linux та MacOS. Godot оптимізований для розробки двовимірних додатків з можливістю створення 3D ігор, однак, дана функція не була достатньо якісно реалізована. Рушій підтримує декілька мов програмування, мову сценаріїв GDScript створену спеціально для потреб двигуна, синтаксис якої подібний до Python, а також мови C#, C++ та VisualScript. Розробники Godot прагнуть зберегти ядро рушія компактним, тому нові функції додаються у вигляді плагінів. Це зменшує необхідність у постійному технічному обслуговуванні коду, який потребує регулярного тестування, та прискорює компіляцію ігор. Менший та структурований код полегшує участь у проекті, а також зменшує розмір файлів редактора.

CryEngine – тривимірний ігровий рушій галузевого рівня, розроблений компанією Crytek для підтримки власних ігор, починаючи з Far Cry та всіх її продовжень [13]. На даний час CryEngine безкоштовний для освітніх цілей, однак для комерційного використання необхідно придбати підписку вартістю 10 доларів США на місяць. Крім того, якщо річний дохід проекту перевищує 5000 доларів, передбачена виплата роялті у розмірі 5%. Рушій підтримує публікацію на платформах Windows, Linux, PlayStation 4, Xbox One, Oculus Rift, OSVR, PSVR та HTC Vive. Незважаючи на те, що рушій має дещо складний інтерфейс редактора, меншу спільноту користувачів та відсутність магазину ресурсів порівняно з іншими популярними ігровими рушіями, CryEngine відомий своєю реалістичною графікою та високою продуктивністю. Сценарії в CryEngine можуть бути написані мовами програмування C++ або Lua.

Unreal Engine 5 – найсучасніша версія одного з перших ігрових рушіїв, розроблених Epic Games [14]. Нещодавно компанія змінила тарифну політику,

дозволивши безкоштовне використання рушія для проектів з річним доходом нижче 1 мільйона доларів та встановила роялті 5% для майбутніх продаж якщо річний оборот більше. Unreal Engine надає доступ до вихідного коду, що значно покращує освітній процес та дозволяє користувачам вносити власні покращення і створювати плагіни. Крім того, рушій має велику спільноту користувачів та ринок ресурсів. Незважаючи, що передусім він був розроблений для підтримки гри від першої особи «Unreal», Unreal Engine став дуже потужним інструментом здатним підтримувати будь-який ігровий жанр, включаючи 2D середовища. Сценарії всередині рушія виконуються за допомогою мови C++ та власної технології візуального програмування Blueprint Visual Scripting, що складається з блоків інтерфейсу на основі вузлів для створення ігрових механік у редакторі. Згідно з документацією, цей підхід дозволяє дизайнерам створювати прототипи та експериментувати без необхідності звертатися до програмістів. Unreal Engine 5 пропонує широкий набір підтримуваних операційних систем, а саме: iOS, Android, Windows, macOS, Linux, а також консолі PlayStation, Xbox та Nintendo Switch. При цьому, перенесення додатку з однієї платформи на іншу не потребує додаткових ресурсів збоку розробників.

Unity – кросплатформений інструмент для розробки відеоігор, що забезпечує можливість портувати кінцевий продук на понад 15 різних операційних систем, включаючи комп’ютери, ігрові консолі, мобільні пристрої, гарнітури доповненої реальності, веб-додатки тощо. Рушій був вперше представлений на всесвітній конференції розробників Apple у 2005 році і відтоді постійно оновлюється [15].

Основними перевагами Unity є наявність візуального середовища розробки, підтримка різних платформ, модульна архітектура компонентів та ціна. Інструмент є безкоштовним для незалежних розробників або компаній з річним доходом до 100 000 доларів. У разі перевищення встановленого порогу необхідно придбати підписку Unity Pro, яка коштує 185 доларів на місяць.

Unity складається з візуального редактора та середовища розробки (IDE). Для написання сценаріїв використовується власна мова UnityScript з синтаксисом

схожим на JavaScript, або мова програмування C#. Редактор Unity складається з вікна проекту, вікна сцени, вікна ієрархії, вікна інспектора, панелі інструментів, вікна консолі та вікон сторонніх плагінів.

Основою архітектури Unity є концепція ігрових об’єктів (GameObject) і компонентів. Ідея полягає в тому, щоб складну механіку можна було зібрати з менших фрагментів логіки. GameObject є базовою сутністю у сцені, яка може бути будь-чим від персонажа до світла або камери. Компоненти додаються до GameObject для надання йому специфічних властивостей і поведінок. Наприклад, компонент Transform визначає положення, обертання та масштаб об’єкта в просторі, тоді як компонент тверде тіло (Rigidbody) дозволяє об’єкту взаємодіяти з фізичним світом. Система сценаріїв у Unity дозволяє програмістам створювати нові компоненти, які можуть керувати поведінкою об’єктів, реагувати на події, обробляти введення користувача, керувати анімацією тощо.

Для роботи Unity Engine використовує різне проміжне програмне забезпечення: DirectX та OpenGL для рендерингу, Beast та Substance для обробки графіки, FMOD для аудіо, PhysX як фізичний рушій, а також RakNet для реалізації мережевих функцій. Додатковим позитивним фактором, що робить Unity привабливим вибором, є велика спільнота користувачів і наявність маркетплейсу, де можна обмінюватися ігровими механіками, компонентами та плагінами

В результаті проведеного огляду було складено таблицю 2.1. На основі отриманих даних був обраний рушій Unity.

Таблиця 2.1 – Порівняння ігрових рушіїв

Назва Рушія	Платформи	Мови програмування	Вартість
Godot	Windows, Linux, MacOS, Android, iOS, PlayStation, XBox, WEB	GDScript, C#, C++, VisualScript	Free

Продовження таблиці 2.1

Назва Рушія	Платформи	Мови програмування	Вартість
CryEngine	Windows, Linux, PlayStation, Xbox One, Oculus Rift, OSVR, PSVR, HTC Vive	C++, Lua.	Free for study, royalty 5%, CryEngine $9.99/міс.
Unreal Engine 5	iOS, Android, Windows, MacOS, Linux, PlayStation, Xbox, Nintendo Switc	C++, Blueprint Visual Scripting	Free, royalty 5%
Unity	iOS, visionOS, Android, Windows, BlackBerry, OSX, Linux, PlayStation, XBox, WEB	UnityScript, C#	Free, Unity Pro $185/міс.

Рішення обрати Unity Enine за снову розробки ґрунтується на таких критеріях:

– Багатоплатформеність. Unity підтримує розробку додатків для багатьох платформ, включаючи ПК, консолі, мобільні пристрої, VR, AR та веб.

– Дружній інтерфейс та легкість у навчанні. Unity пропонує інтуїтивно зрозумілий інтерфейс, що робить його привабливим для новачків. Наявність великої кількості навчальних матеріалів, форумів, відеоуроків та документації, сприяє швидкому вирішенню проблем та обміну знаннями.

– Asset Store. Магазин ресурсів Unity Asset Store пропонує безліч готових моделей, скриптів, звуків та інших активів.

– Безкоштовний доступ. Unity має безкоштовну версію з великим набором інструментві, що є важливим фактором для інді-розробників та маленьких студій.

– Модульність та інтеграції. Unity gідтримує інтеграцію інструментів та служб таких як Visual Studio, Firebase, AdMob та інші.

– Висока продуктивність. Unity забезпечує високу продуктивність та оптимізацію завдяки можливостям для налаштування.

Завдяки своїй гнучкості Unity дозволяє створювати унікальні функціональні можливості, адаптовані під конкретні потреби.

2.2 Вибір засобів розробки AR

Здійснено порівняльний аналіз найвідоміших бібліотек доповненої реальності: ARToolKit, Wikitude, ARKit, ARCore, EasyAR, Kudan, Vuforia, з метою визначення найоптимальнішого SDK для поточної розробки.

SDK (Software Development Kit) – це набір інструментів, документації, бібліотек та інших ресурсів, що надаються розробникам для створення програмного забезпечення для певної платформи, операційної системи або середовища. Використання SDK дозволяє розробникам застосовувати стандартизовані та перевірені рішення і шаблони під час розробки, що зменшує необхідність індивідуального вирішення типових задач, тим самим оптимізуючи час та ресурси, затрачені на розробку.

Поширеними бібліотеками, що мають широке застосування є:

ARToolKit – це бібліотека доповненої реальності на основі маркерів [16]. Бібліотека складається з модуля AR, який містить процедури для відстеження двомірних міток, калібрування та збір вхідних даних відеокамери, а також програмного інтерфейсу OpenGL, що реалізує кінцеву візуалізацію. OpenGL – специфікація, що визначає незалежний від мови програмування інтерфейс для розробки додатків, що використовують двовимірну або тривімірну комп’ютерну графіку. Основними перевагами SDK є відкритий вихідний код та відсутність WaterMark (водяний знак). WaterMark – це напівпрозорий малюнок, текст або графічний елемент, що накладається поверх зображення камери. З метою захисту авторських прав, підтвердження автентичності або маркування власності. ARToolKit підтримує декілька платформ, зокрема Linux, MacOS, Windows, Android, iOS.

Wikitude – набір засобів для розробки додатків доповненої реальності, який забезпечує можливість розпізнавання та відстеження зображень і об’єктів

реального середовища для використання їх в якості маркерів та функцію розширеного відстеження (Extended Tracking), яка дозволяє зберігати положення мітки, навіть коли вона виходить за межі огляду камери [17]. Бібліотека базується на веб-технологіях, таких як HTML, JavaScript і CSS, що дає змогу створювати кросплатформенні AR-додатки. На даний час, SDK підтримує платформи Android та iOS.

ARKit – платформа доповненої реальності, розроблена компанією Apple, яка надає розробникам можливість створювати додатки з елементами AR виключно для пристроїв на базі iOS [18]. ARKit автоматично визначає горизонтальні та вертикальні площини, аналізує навколишнє середовище, адаптуючи освітлення віртуальних об’єктів відповідно до реальних умов. Для пристроїв, оснащених камерою TrueDepth, таких як iPhone X, ARKit може відстежувати вирази обличчя та рухи голови, що дозволяє створювати анімовані аватари та маски. TrueDepth – технологія для точного відображення геометрії обличчя, яка працює аналізуючи тисячі невидимих точок, щоб створити цифрову карту глибини та інфрачервоне зображення обличчя.

ARCore – набір інструментів для створення AR-застосунків, розроблений корпорацією Google [19]. Ключовими функціями ARCore є відстеження руху (Motion Tracking), визначення площин (Environmental Understanding), оцінка освітлення (Light Estimation) та Cloud Anchors. Функція Cloud Anchors дозволяє створювати спільний AR-досвід між різними пристроями, дозволяючи кільком користувачам взаємодіяти з одним і тим ж віртуальним об’єктом. Перевагами ARCore є його доступність, інтеграція з іншими технологіями Google, такими як Google Maps та активна підтримка і регулярні оновлення.

EasyAR – проста у використанні бібліотекою, яка підтримує розпізнавання лише 2D об’єктів [20]. Платформа представлена кількома версіями: EasyAR Sense, що включає всі основні функції AR, EasyAR Cloud, що підтримує хмарні обчислення для розпізнавання зображень та EasyAR CRS (Cloud Recognition Service), спеціалізований сервіс для хмарного розпізнавання зображень, що забезпечує високу швидкість і точність. Серед переваг EasyAR SDK можна

відзначити легкість інтеграції завдяки простому у використанні API, високу продуктивність та гнучкість, що дозволяє використовувати її як у простих, так і у складних інтерактивних додатках з 3D-графікою.

Kudan – це програмний пакет, розроблений для створення додатків доповненої реальності, який забезпечує розпізнавання та відстеження 2D і 3D об’єктів на основі власної технології комп’ютерного зору [21]. Основна особливість Kudan SDK полягає в здатності відстежувати положення та орієнтацію об’єктів у реальному часі без використання додаткових датчиків глибини чи стереокамер, завдяки захопленню характерних точок зображення та обчислення перспективи з шістьма ступенями свободи. SDK підтримує як розпізнавання маркерів, так і безмаркерну роботу, не обмежує кількість розпізнаваних зображень та займає невеликий обсяг пам’яті.

Vuforia – передова платформа доповненої реальності, яка дозволяє інтегрувати віртуальний контент у фізичне середовище [22]. Основним компонентом SDK є Vuforia Engine, який забезпечує функціонування технологій комп’ютерного зору та реконструкції навколишнього середовища (Smart Terrain). Vuforia підтримує різні типи міток: 2D, 3D, багатоцільові конфігурації, циліндричні мішені для відстеження зображень на циліндричних поверхнях, а також зображення без маркера та текстові мітки. Крім того, Vuforia пропонує VuMark, настроюваний візуальний код, що поєднує в собі зображення і QR-код для вказання доступного AR-досвіду. Рушій підтримує розпізнавання об’єктів з використанням локальної або хмарної бази даних, що дозволяє відстежувати до одного мільйона цілей одночасно, Extended Tracking та Virtual Button – технологія взаємодії з віртуальною інформаціює шляхом фізичного натискання на відповідні області віртуальних об’єктів.

На основі проведеного порівняльного аналізу найвідоміших бібліотек доповненої реальності було складено таблицю 2.2. В результаті аналізу було обрано Vuforia як оптимальний інструмент для розробки.

Таблиця 2.2 – Порівняння AR SDK

Назва SDK	Тип ліцензії	Платформи	Тип маркерів	Технології	Підтримка Unity
ARToolKit	Free Open Source	Android, iOS, Linux, Windows, maxOS	2D	Computer vision	+
Wikitude	Commercial	Android, iOS	2D,3D	SLAM, Computer vision, Extended Tracking, Geo AR, Cloud Recognition	+
ARKit	Free	iOS	Face	SLAM, Computer vision	+
ARCore	Free, Commercial	Android, iOS, Windows	2D,3D	SLAM, Computer vision, Geo AR,	+
EasyAR	Free, Commercial	Android, iOS, Windows, maxOS	2D	SLAM, Computer vision, Cloud Recognition	+
Kudan	Free, Commercial	Android, iOS	2D,3D	SLAM, Computer vision, Cloud Recognition	+
Vuforia	Free, Commercial	Android, iOS, UPW	2D,3D, VuMark	Computer vision, Extended Tracking, Geo AR, Cloud Recognition	+

Vuforia є однією з найпопулярніших платформ, що забезпечує легкий доступ до великої кількості навчальних матеріалів. Бібліотека пропонує розробникам повний набір інструментів для створення AR-додатків, а саме:

– розпізнавання різного роду маркерів 2D, 3D, VuMark;

– розпізнавання декількох цілей одночасно, включаючи об’єкти зображення і текст;

– підтримка технології Extended Tracking;

– віртуальні кнопки;
– геолокаційна AR;
– відображення додаткових елементів через OpenGL;
– підтримка технології Smart Terrain;
– хмарне розпізнавання (Cloud Recognition).

Однією з ключових переваг Vuforia є підтримка технології Cloud Recognition що доволяє використовувати дані, які знаходяться як на пристрої, так і в хмарі. Також платформа підтримує пристрої віртуальної реальності, що розширює можливості її використання. Однак, недоліком є відсутність структурованої документації, що ускладнює перші кроки у роботі з SDK.

Таким чином, Vuforia є оптимальним вибором для розробки додатку доповненої реальності завдяки потужним можливостям розпізнавання та відстеження, широкій підтримці різних типів міток, інтеграції з хмарними сервісами. Вибір цього SDK дозволяє створити багатофункціональний та гнучкий додаток, який відповідає поставленим потребам та забезпечує високу якість взаємодії з доповненою реальністю.

РОЗДІЛ 3
РОЗРОБКА AR-ДОДАТКУ

3.1Постановка задачі

Метою даної кваліфікаційної роботи є розробка додатку-помічника для працівників виробництва, що спеціалізується на складанні багатокомпонентних виробів. Для реалізації поставленої задачі функціонал додатку повинен містити наступні компоненти:

– Механізм динамічного завантаження даних, оскільки специфіка роботи рушія Unity полягає у тому, що об’єкти, розміщені на сцені при запуску застосунку, завантажуються в оперативну пам’ять. Завантаження всіх даних одразу призведе до нераціонального використання ресурсів пристрою.

– Простий та інтуїтивно зрозумілий інтерфейс, який не викликатиме труднощів у використанні та не вимагатиме додаткового навчання персоналу.

– Сканер QR-кодів для уникнення помилкового підбору деталей для складання, оскільки в деяких випадках каталожні номери різних комплектуючих відрізняються лише кількома цифрами.

– Систему мультимодального введення, яка підтримує різні типи пристроїв, включаючи сенсорні екрани, стандартні засоби введення (мишка, клавіатура), та безконтактний метод взаємодії за допомогою відстеження положення та жестів руки.

– Технологію проектування тривимірних інструкцій, що доповнюють реальне середовище віртуальною інформацією, наочно демонструючи алгоритм дій, який необхідно виконати для складання певного виробу.

3.2 Реалізація механізму завантаження даних

Для реалізації завантаження збережених даних, які знаходяться у файлі JsonData, була розроблена система завантаження, що складається з трьох класів: JsonProductLoader, JsonProductData та JsonDetailData.

Структура JsonProductData є загальною для продукту та містить інформацію про ідентифікатор, посилання на 3D-модель з анімаціями та список об’єктів типу JsonDetailData для кожного виду виробу. Структура JsonDetailData призначена для зберігання даних про деталі, що входять до складу продукту, і містить інформацію про каталожний номер, загальний опис, додаткові текстові інструкції та посилання на зображення користувацького інтерфейсу.

Безпосередньо за ініціалізацію даних відповідає клас JsonProductLoader, який асинхронно завантажує дані, маючи посилання на розміщення файлу JsonData. Сценарій десеріалізує зашифровані дані назад у змінні оперативної пам’яті для їх подальшого використання. Десеріалізація – це процес, зворотний до серіалізації, який передбачає отримання потоку байтів і перетворення його назад в об’єкт. Програмний код методу асинхронного завантаження продемонстрований у лістингу 3.1.

Лістинг 3.1 – Програмний код функції «LoadData»

```
        private async UniTask LoadData(string fileName)
        {
            string url = string.Empty;
            url = "file://" + Application.dataPath + "/Resources/RemoteConfigs/" +
fileName;
            UnityWebRequest request = UnityWebRequest.Get(url);
            await request.SendWebRequest();
            if (request.result == UnityWebRequest.Result.ConnectionError ||
                request.result == UnityWebRequest.Result.ProtocolError)
            {
                Debug.LogError(request.error);
            }
            else
            {
                var text = request.downloadHandler.text;
                _remoteProductData =
JsonConvert.DeserializeObject<List<JsonProductData>>(text);
                ConvertProductData();
                _isLoaded = true;
            }
        }
```

Кінець лістингу 3.1

Для підтримки асинхронного завантаженя даних був використаний пакет інструментів UniTask. UniTask – популярна бібліотека, яка працює без втручання SynchronizationContext використовуючи новий тип легкої задачі UniTask, що

працює напряму з PlayerLoop API. PlayerLoop – це клас взаємодії з циклом оновлення ядра Unity. Функції класу, дозволяють отримати інформацію про порядок оновлення систем двигуна та встановити власний [23].

SynchronizationContext – базовий клас, що надає контекст з вільним потоком без вбудованої синхронізації. Принцип його роботи полягає в тому, що завдання розміщуються в черзі контексту, а не конкретного потоку.

Проблема SynchronizationContext у середовищі Unity полягає в тому, що при використанні стандартних засобів, таких як «await Task.Yield();», для примусового асинхронного завершення методу, залишок виконання методу відправляється в контекст, який встановлює пріоритети для цієї роботи відносно інших завдань, що очікують виконання. У більшості випадків цей підхід призводить до того, що пріоритет надається завданням розміщеним у контексті вище, а не задачам відстеження введення та рендерингу, що в свою чергу знижує швидкість роботи інтерфейсу.

3.3 Реалізація інтерфейсу користувача

Для реалізації інтерфейсу користувача було використано бібліотеку з відкритим вихідним кодом View Manager [24]. View Manager забезпечує готове ядро графічного інтерфейсу, підтримку потокового використання ресурсів, префабів та ScriptableObject.

Потокові ресурси (Streaming assets) в Unity – це будь-які ресурси, зокрема зображення, аудіо, відео, скрипти, текстові файли, які зберігаються в спеціальній папці «Resources» та завантажуються Unity Player за потреби під час роботи додатку. Механізм забезпечує ефективне управління ресурсами, дозволяючи зменшити використання оперативної пам’яті, оскільки ресурси завантажуються лише тоді, коли вони необхідні.

Prefab – шаблон, який дозволяє створювати, налаштовувати та зберігати GameObject разом із усіма його компонентами, значеннями властивостей і дочірніми об’єктами. Prefab Asset формує багаторазовий ресурс, на основі якого

можна створювати нові екземпляри Prefab на сцені. Встановлений ієрархічний зв’язок значно спрощує процес контролю над екземплярами, оскільки зміни в шаблоні впливають на всі похідні об’єкти. Тому для створення візуальної частини інтерфейсу було використано систему префабів. Для створення власного вікна необхідно сконструювати його зовнішній вигляд за допомогою інструментів редактора Unity та зберегти у файлах проекту як префаб. На рисунку 3.1 зображено зовнішній вигляд інтерфейсу додатка.

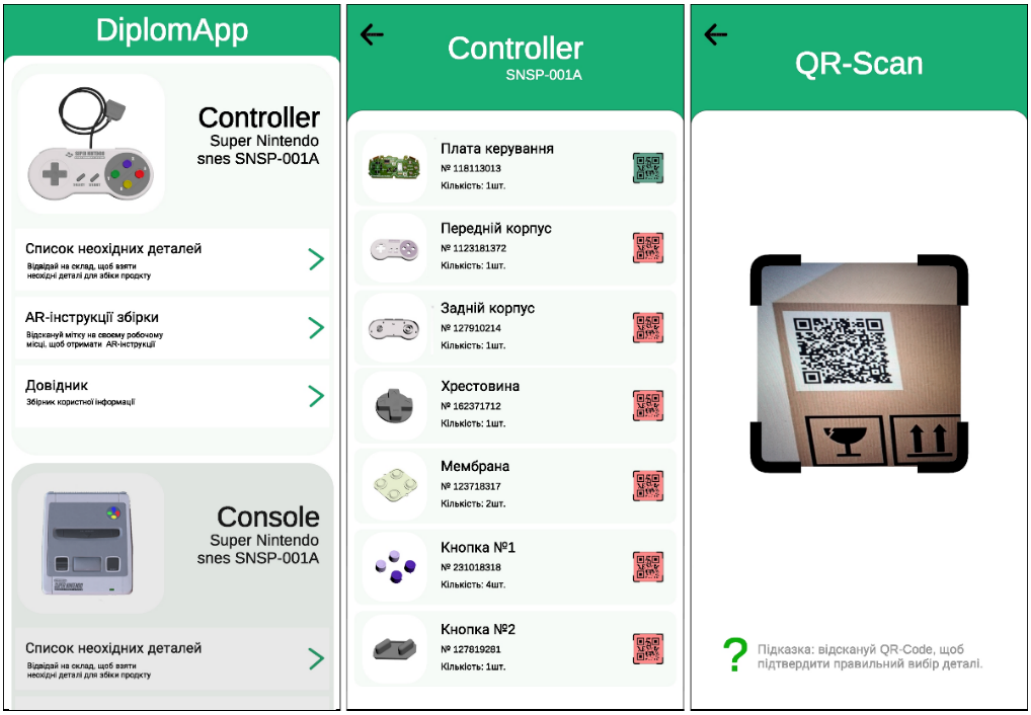


Рисунок 3.1 – Інтерфейс користувача

Для зберігання інформації про всі наявні вікна застосунку був використаний ScriptableObject. ScriptableObject – контейнер, призначений для зберігання значних обсягів інформації незалежно від екземпляру класу. Принцип роботи механізму полягає в оптимізації використання пам’яті в проекті шляхом уникнення дублювання даних. Зберігаючи унікальні значення в одному місці, ScriptableObject надає можливість спільно використовувати їх різним об’єктам, що знижує загальний об’єм задіяної пам’яті.

Реалізація поведінкової логіки інтерфейсу була створена на основі екземпляру вікна під назвою MainView, що був інтегрований до відповідного об’єкту в ієрархії сцени. Бібліотека ViewManager надає кілька базових функцій керування інтерфейсом: методи «AddView» та «RemoveView» додають і видаляють вікно з полотна на визначеному шарі. Методи «RemoveAllViews» та «RemoveAllDialogs» призначені для знищення всіх об’єктів View та Dialog. Метод «IsViewActive» повертає значення true, якщо перегляд активний. Методи «EnableInput» та «DisableInput» дозволяють керувати можливістю введення для всіх об’єктів, що контролює ViewManager. Комбінуючи ці методи, була створена логіка поведінки графічного інтерфейсу. Наприклад, при натисканні кнопки «Назад» закриваються всі активні вікна та відкривається останнє активне, тобто MainView. Програмний код реалізації наведено у лістингу 3.2.

Лістинг 3.2 – Програмний код функції «BackButtonClick»

```
public void BackButtonClick()
{
    ViewManager.Instance.RemoveAllViews();
    ViewManager.Instance.AddView(View.MainView);
}
```

Кінець лістингу 3.2

3.4 Реалізація сканера QR-кодів

Алгоритм використання сканера полягає в наступному: працівник з пристроєм вирушає на склад і відкриває список необхідних комплектуючих. Знайшовши відповідну коробку, він повинен відсканувати QR-код для перевірки правильності вибору. У разі вірного вибору, піктограма QR-коду змінює колір на зелений (рис. 3.2). В іншому випадку додаток пропонує перевірити свій вибір.

Рисунок 3.2 – Підтвердження вибору ситемою QR-Scan

QR-код сканер функціонує як автономна окрема система. Процес декодування зображення здійснюється за допомогою бібліотеки «zxing.dll». Для відображення зображення використовується WebCamTexture, тип текстур, що візуалізують відеопотік у режимі реального часу. Програмний код реалізації продемонстрований у лістингу 3.3.

Лістинг 3.3 – Програмний код функції «GetQRCode»

```
    IEnumerator GetQRCode()
    {
        IBarcodeReader barCodeReader = new BarcodeReader();
        webcamTexture.Play();
        var snap = new Texture2D(webcamTexture.width, webcamTexture.height,
TextureFormat.ARGB32, false);
        QrCode = string.Empty;

        while (string.IsNullOrEmpty(QrCode))
        {
            try
            {
                snap.SetPixels32(webcamTexture.GetPixels32());
                var Result = barCodeReader.Decode(snap.GetRawTextureData(),
webcamTexture.width, webcamTexture.height, RGBLuminanceSource.BitmapFormat.ARGB32);
                if (Result != null)
                {
                    QrCode = Result.Text;
                    if (!string.IsNullOrEmpty(QrCode))
                    {
                        _eventBus.Invoke(new QRCodeSignal(QrCode));
                        break;
                    }
                }
            }
            catch (Exception ex) { Debug.LogWarning(ex.Message); }
            yield return null;
        }
            webcamTexture.Stop();
    }
```

Кінець лістингу 3.3

3.5 Реалізація системи мультимодального введення

Для реалізації системи мультимодального введення була розроблена невелика бібліотека за допомогою засобів фреймворку ManoMotion. ManoMotion – набір інструментів призначений для інтеграції технології розпізнавання жестів рук у мобільні та інші програмні продукти. SDK використовує комп'ютерний зір та алгоритми машинного навчання для відстеження рухів рук у режимі реального часу, що надає можливість створити інтуїтивно зрозумілий інтерфейс користувача [25].

Як результат роботи, ManoMotion повертає ключові точки долоні та кожного пальця, розмір і положення руки. Жести в свою чергу поділяються на два типи: TriggerGestures та ContinuousGestures. TriggerGestures – різновид жестів, що представлені певною послідовністю рухів руки та пальців. Наприклад, швидкий дотик вказівного та великого пальців означає клік. Другий тип, Continuous Gestures – відбувається у разі неперервної демонстрації одного виду жесту. Наприклад, утримання та переміщення чогось, означає Hold, тобто імітацію тримання.

Дана інформація та інформація з попередніх кадрів, дозволяє визначити тип жесту, який показує користувач. Наприклад, якщо при типі TriggerGestures, поле State у нульовому кадрі дорівнює 4, а наступному 10, логіка коду може припустити, що користувач стиснув кулак. Поле State вказує на ступінь відкритості або закритості руки, присвоюючи стану числове значення від 0 до 13, де 0 означає повністю відкриту руку, а 13 повністю закриту.

Для реалізації власного жесту було створено користувацький тип даних HandEvent. Структура HandEvent складається з підструктури BoundingBox, яка містить інформацію про розміри та положення руки відносно камери. Підструктура HandPoints містить інформацію про просторове положення кожного з пальців, а також двох перелічуваних типів даних Trigger та Continuous, які представляють основні типи жестів та містять свої підвиди. (лістинг 3.4).

Лістинг 3.4 – Структура «HandEvent»

```
public struct HandEvent
{
    public BoundingBox BoundingBox;
    public HandPoints HandPoints;
    public GestureTrigger Action;
    public GestureContinuous Continuous;
    public float Depth;

    public HandEvent(BoundingBox boundingBox, HandPoints handPoints,
        GestureTrigger action, GestureContinuous continuous, float depth = 0)
    {
        BoundingBox = boundingBox;
        HandPoints = handPoints;
        Action = action;
        Continuous = continuous;
        Depth = depth;
    }
}
```

Кінець лістингу 3.4

Для контролю над отриманими даними створено клас HandDriver, який відповідає за кешування інформації в оперативній пам’яті та додає її до масиву нещодавніх станів руки. Основна реалізація логіки жестів виконується у класах HandSwipeHandler, HandClickHandler та HandDragHandler, які є нащадками абстрактного класу InputController. Базовий клас містить абстрактні методи OnActivate, OnDeactivate, OnProcess, булеву змінну IsActive та цілочислену Priority. Методи стану викликаються при активації та деактивації контролера і використовуються для встановлення необхідних параметрів. Функція OnProcess викликається в кожному кадрі й використовується для проведення необхідних обчислень.

Клас HandSwipeHandler реалізує жест «Swipe», тобто пролистування. Використовуючи масив, створений класом HandDriver, можна отримати інформацію про положення руки в кожному кадрі, що дозволяє створити послідовність станів. Послідовність використовується для визначення напрямку та величини зміщення центру руки між кадрами. Якщо зміщення перевищує встановлену похибку та направлене в один бік, викликається подія «OnSwipe». Програмний код реалізації методу «SwipeEventHandler» наведений у лістингу 3.5.

Лістинг 3.5 – Програмний код функції «SwipeEventHandler»

```
    public void SwipeEventHandler(Vector2 delta)
    {
        if (Mathf.Abs(delta.x) > swipeThreshold || Mathf.Abs(delta.y) >
swipeThreshold)
        {
            if (!isSwiping)
            {
                isSwiping = true;
                if (Mathf.Abs(delta.x) > Mathf.Abs(delta.y))
                {
                    if (delta.x > 0)
                    {
                        OnSwipeRight();
                    }
                    else
                    {
                        OnSwipeLeft();
                    }
                }
                else
                {
                    if (delta.y > 0)
                    {
                        OnSwipeUp();
                    }
                    else
                    {
                        OnSwipeDown();
                    }
                }
            }
        }
        else   {
            isSwiping = false;
        }
    }
```

Кінець лістингу 3.5

Для підтримки мультимодального введення реалізовано TouchDriver та MouseKeyboardDriver, які розроблені за аналогічним принципом та відповідають за введення з сенсорного екрану та миші з клавіатурою відповідно. Управління системою введення здійснює InputManager через інтерфейс IInputDriver, який містить методи ініціалізації та деініціалізації драйверів залежно від умов та платформи. Структура бібліотеки зображена в додатку А.

3.6 Розробка прототипу додатку

Оскільки додаток побудований за принципами модульної архітектурі для обміну даними між системами застосунку використовується шина подій. Шина

подій (Event Bus) – це патерн проектування, за допомогою якого компоненти можуть надсилати та отримувати повідомлення. Шина функціонує як посередник між компонентами, надаючи можливість обмінюватися повідомленнями без необхідності явного з'єднання між ними. Реалізація Event Bus представлена в класі CustomEventBus, який знаходиться в додатку В. Передача інформації здійснюється за допомогою сигналів. Сигнал – це екземпляр класу, який знаходиться в просторі імен CustomEventBus.Signals і реалізує тип даних, який необхідно передати.

Наприклад, сканер QR-кодів працює як незалежний компонент і в результаті своєї роботи генерує дані типу string, які необхідно передати в сценарій PartsManager для перевірки на збіг з деталлю зі списку. Взаємодія відбувається за допомогою сигналу QRCodeSignal, що має поле Value типу string, та конструктор класу, який присвоює значення змінній. Виклик події відбувається через метод Invoke, в який передається новий екземпляр сигналу зі змінною. Щоб стати слухачем, необхідно підписатися на подію в стандартному методі «Start», який спрацьовує під час ініціалізації сценарію. Підписка здійснюється за допомогою функції Subscribe, вказавши в кутових дужках необхідний сигнал.

Застосунок розроблено на основі технології маркерної доповненої реальності (Marker-based AR). Тому необхідною умовою для додавання віртуального вмісту в реальне середовище є створення маркеру. Для цього Vuforia надає функцій офіційного сайту. В розділі для розробників «Target Manager» була створена база даних під назвою «InstructionMarker». Для наповнення бази даних було завантажене відповідне зображення (рис. 3.3), яке відповідає вимогам, контрастності та рівню деталізації, щоб забезпечити оптимальне розпізнавання камерою пристрою.

Рисунок 3.3 – Маркер доповненої реальності

Для роботи з доповненою реальністю необхідно імпортувати базу даних до файлів проекту. Процес створення AR реалізується за допомогою компонента «ImageTarget», який розміщується на сцені та присвоюється зображення з бази даних. Компонент «ImageTarget» виконує роль орієнтиру для прив’язки цифрової інформації, віртуальний контент розміщується як дочірній об’єкт маркера в ієрархії сцени. Це забезпечує відображення інформації відповідно до положення та орієнтації маркера в просторі.

Для демонстрації роботи прототипу застосунку було завантажено CAD-модель контролера ігрової приставки Super Nintendo Entertainment System (SNSP-001A) [26]. Тривимірна модель зображена на рисунку 3.4.

Рисунок 3.4 – Завантажена 3D модель

Для наочності процес відображення інструкцій відбувається з використанням анімації та текстових підказок. Анімації дозволяють користувачу краще зрозуміти дії, які необхідно виконати для досягнення результату, тоді як текстові інструкції детально описують послідовність дій які неохідно виконати. Робота додатку зображена на рисунку 3.5.

Рисунок 3.5 – Демонстрація роботи додатку

ВИСНОВКИ

У ході дослідження було розроблено кросплатформений AR-застосунок в середовищі Unity. Проведено всебічний огляд основних типів та технологій для побудови доповненої реальності, зокрема: Marker-based AR, Marker-less AR, Location-based AR, Projection-based AR. Детально проаналізовано існуючі бібліотеки для створення AR-додатків, такі як ARToolKit, Wikitude, ARKit, ARCore, EasyAR, Kudan, Vuforia, а також середовища розробки Godot, CryEngine, Unreal, Unity.

Розроблений застосунок можна використовувати для оптимізації виробничих процесів та покращення навчальних програм. Впроваджений жестовий інтерфейс користувача дозволяє взаємодіяти з цифровими об’єктами на інтуїтивному рівні, покращуючи занурення в віртуальне середовище. Реалізований сканер QR-кодів, дозволяє зменшити кількість помилок персоналу у випадках неправильного підбору комплектуючих для збирання. Маркерне орієнтування 3D-інструкцій у просторі дозволяє отримувати інформацію на більш інтуїтивному рівні, в передбачному місці.

На основі проведених тестувань та налаштувань застосунку встановлено, що даний AR-застосунок здатний значно оптимізувати виробничі процеси шляхом інтеграції необхідних інструкцій та елементів у реальне середовище, наочно демонструючи, що необхідно зробити. Застосування доповненої реальності забезпечує наочність і доступність інформації, що сприяє підвищенню ефективності виробничих та навчальних процесів, знижуючи ймовірність помилок та покращуючи загальний рівень продуктивності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Cizmeci D. Augmented reality examples how businesses are utilising AR. Daglar Cizmeci. URL: https://daglar-cizmeci.com/augmented-reality-examples-how-businesses-are-utilising-ar/ (дата звернення: 21.05.2024).

2. Leopold G. Lockheed martin embraces AR on the shop floor. EETimes. URL: https://www.eetimes.com/lockheed-martin-embraces-ar-on-the-shop-floor/ (дата звернення: 21.05.2024).

3. What is Industry 4.0 and how does it work?. IBM in Deutschland. URL: https://www.ibm.com/topics/industry-4-0 (дата звернення: 21.05.2024).

4. Howarth J. 24 augmented reality stats. Exploding Topics. URL: https://explodingtopics.com/blog/augmented-reality-stats (дата звернення: 21.05.2024).

5. How can augmented reality experience the way classroom learning. Augray Blog. URL: https://www.augray.com/blog/how-can-augmented-reality-change-the-way-classroom-learning/ (дата звернення: 21.05.2024).

6. Zvejnieks G. How to create 3D content in Blender for mobile AR - Overlyapp. Overlyapp - Overlyapp. URL: https://overlyapp.com/blog/how-to-create-3d-content-in-blender-for-mobile-augmented-reality-projects/ (дата звернення: 21.05.2024).

7. Sutherland I. The sword of damocles. URL: https://www.researchgate.net/figure/The-Sword-of-Damocles-by-Ivan-Sutherland_fig2_291516650 (дата звернення: 22.05.2024).

8. Analysis of the use of VR in medical and science and technology. ResearchGate. URL: https://www.researchgate.net/figure/Exploration-of-airflow-using-Virtual-Wind-Tunnel-developed-at-NASA-Ames-7-8_fig1_365340796 (дата звернення: 22.05.2024).

9. Ho T. Principle of operation of Augmented Reality - ITZone. ITZone. URL: https://itzone.com.vn/en/article/principle-of-operation-of-augmented-reality/ (дата звернення: 22.05.2024).

10. Троянов С. Історія Pokemon GO: як змусити інтровертів вийти на вулицю та почати спілкуватись. Na chasi. URL: https://nachasi.com/videogames/2020/10/07/pokemon-go/ (дата звернення: 22.05.2024).

11. Annand G. The history of the id tech engine. SUPERJUMP. URL: https://www.superjumpmagazine.com/the-history-of-the-id-tech-engine/ (дата звернення: 22.05.2024).

12. Godot. Godot engine. Версія 4.2. URL: https://docs.godotengine.org/en/stable/ (дата звернення: 22.05.2024).

13. Cryengine. URL: https://www.cryengine.com/tutorials (дата звернення: 22.05.2024).

14. Unreal engine. Версія 5.0. URL: https://dev.epicgames.com/documentation/ru-ru/unreal-engine/unreal-engine-5-0-documentation?application_version=5.0 (дата звернення: 23.05.2024).

15. Unity documentation. Unity. URL: https://docs.unity.com/ (дата звернення: 23.05.2024).

16. ARToolKit documentation. artoolkit-docs. URL: https://kalwalt.github.io/artoolkit-docs/ (дата звернення: 23.05.2024).

17. Wikitude SDK documentation. Wikitude. URL: https://www.wikitude.com/external/doc/documentation/ (дата звернення: 23.05.2024).

18. ARKit apple developer documentation. Apple. URL: https://developer.apple.com/documentation/arkit (дата звернення: 24.05.2024).

19. Overview of ARCore and supported development environments. Google for Developers. URL: https://developers.google.com/ar/develop (дата звернення: 24.05.2024).

20. EasyAR developer center. EasyAR. URL: https://www.easyar.com/view/support.html (дата звернення: 24.05.2024).

21. The kudan developer hub. Kudan AR SDK. URL: https://www.xlsoft.com/doc/kudan/ (дата звернення: 24.05.2024).

22. Vuforia library. Engine Developer Portal. URL: https://developer.vuforia.com/library/ (дата звернення: 25.05.2024)

23. UniTask library. GitHub. URL: https://github.com/Cysharp/UniTask (дата звернення: 25.05.2024)

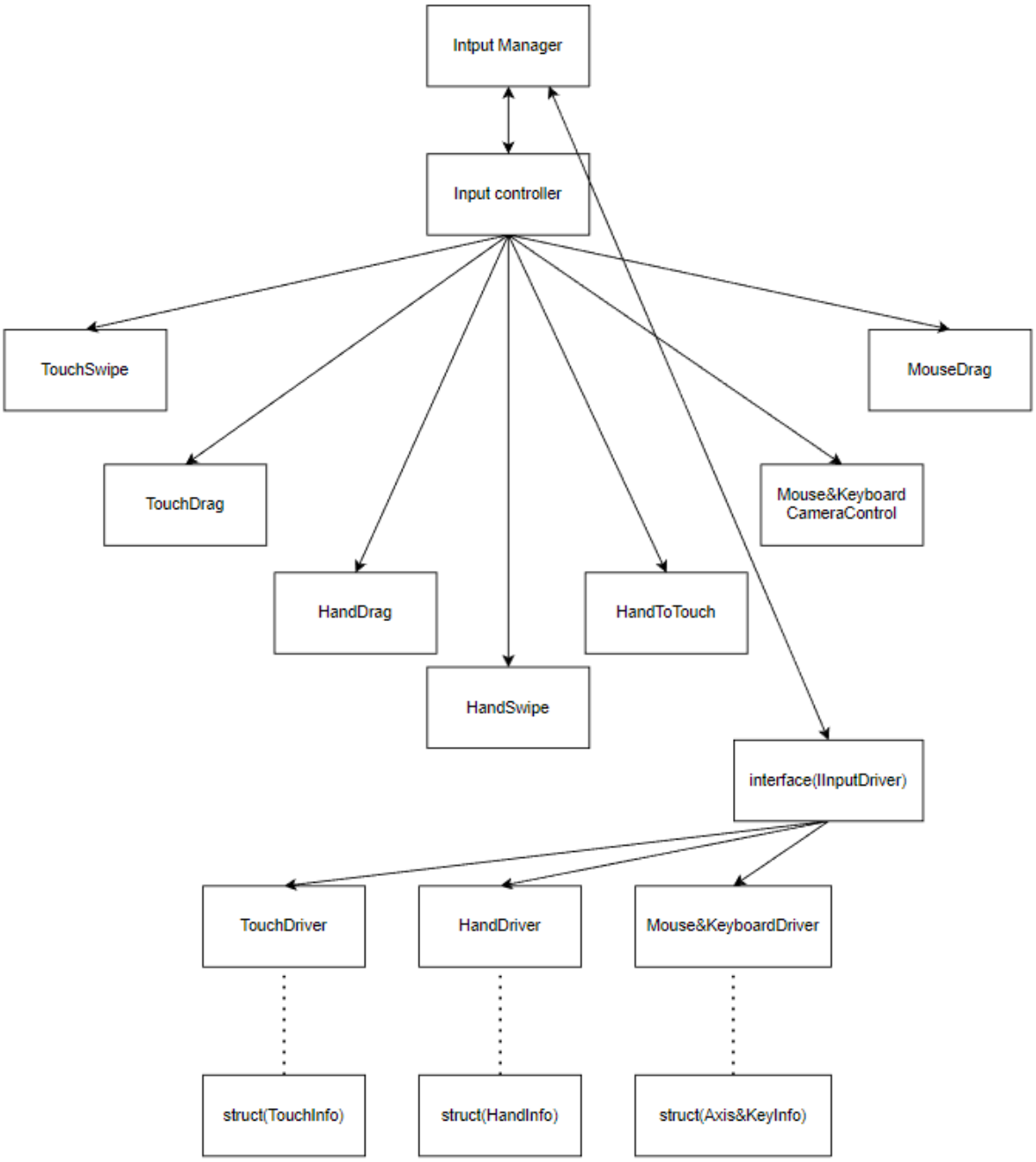
24. View Manager library. GitHub. URL: https://github.com/jonHuffman/ViewManager (дата звернення: 25.05.2024)

25. ManoMotion library. ManoMotion API documentation. URL: https://www.sdk.manomotion.com/SDK_Pro_v1.4.4.1/index.html (дата звернення: 25.05.2024)

26. CAD-model. Sketchfab Store. URL: https://sketchfab.com/3d-models/nintendo-controller-3a56407501f746f4af6348f3174c144d (дата звернення: 25.05.2024)

ДОДАТКИ

Додаток А
Структура системи введення



Додаток Б
Реалізація патерну EventBus

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

namespace CustomEventBus
{
    public class EventBus : IService
    {
        private Dictionary<string, List<CallbackWithPriority>> _signalCallbacks = new
Dictionary<string, List<CallbackWithPriority>>();

        public void Subscribe<T>(Action<T> callback, int priority = 0)
        {
            string key = typeof(T).Name;
            if (_signalCallbacks.ContainsKey(key))
            {
                _signalCallbacks[key].Add(new CallbackWithPriority(priority,
callback));
            }
            else
            {
                _signalCallbacks.Add(key, new List<CallbackWithPriority>() { new
(priority, callback) });
            }

            _signalCallbacks[key] = _signalCallbacks[key].OrderByDescending(x =>
x.Priority).ToList();
        }

        public void Invoke<T>(T signal)
        {
            string key = typeof(T).Name;
            if (_signalCallbacks.ContainsKey(key))
            {
                foreach (var obj in _signalCallbacks[key])
                {
                    var callback = obj.Callback as Action<T>;
                    callback?.Invoke(signal);
                }
            }
        }

        public void Unsubscribe<T>(Action<T> callback)
        {
            string key = typeof(T).Name;
            if (_signalCallbacks.ContainsKey(key))
            {
                var callbackToDelete = _signalCallbacks[key].FirstOrDefault(x =>
x.Callback.Equals(callback));
                if (callbackToDelete != null)
                {
                    _signalCallbacks[key].Remove(callbackToDelete);
                }
            }
        }
    }
}
```

Додаток В
Реалізація патерну Singleton

```
using UnityEngine;

public class Singleton<T> : MonoBehaviour where T : Singleton<T>
{
	public static T Instance
	{
		get
		{
			if (instance == null)
			{
				T[] managers = Object.FindObjectsOfType(typeof(T)) as T[];
				if (managers.Length != 0)
				{
					if (managers.Length == 1)
					{
						instance = managers[0];
						instance.gameObject.name = typeof(T).Name;
						return instance;
					}
					else
					{
						foreach (T manager in managers)
						{
							Destroy(manager.gameObject);
						}
					}
				}
				var go = new GameObject(typeof(T).Name, typeof(T));
				instance = go.GetComponent<T>();
				DontDestroyOnLoad(go);
			}
			return instance;
		}
		set
		{
			instance = value as T;
		}
	}
	private static T instance;
}
```