Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

магістра
(назва освітньо-кваліфікаційного рівня)

студента	*Тріско Антона Євгеновича*
	(ПІБ)
академічної групи	*122М-22-2*
	(шифр)
спеціальності	*122 Комп'ютерні науки*
	(код і назва спеціальності)
освітньої програми	*«Комп'ютерні науки»*
	(назва освітньої програми)
на тему:	*Дослідження способів та принципів взаємодії з Artificial Reality на прикладі ігрового додатку*

_____*А.Є. Тріско*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
розділів кваліфікаційної роботи	*доц. Приходченко С.Д.*			
спеціальний	*доц. Приходченко С.Д.*			
економічний				
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2023

Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
Завідувач кафедри
Програмного забезпечення комп'ютерних систем
(повна назва)

_____ М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » _____ 20 23 Року

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

студенту *122м-22-2* *Тріско Антону Євгеновичу*
(група) (прізвище та ініціали)

Тема кваліфікаційної роботи *Дослідження способів та принципів взаємодії з Artificial Reality на прикладі ігрового додатку*

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від 31.10.2023 р. № 1200-с

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень – взаємодія з Artificial Reality (AR) у контексті ігрового додатку.

Предмет досліджень – методи та принципи взаємодії користувача з AR-середовищем в ігровому додатку.

Мета НДР – підвищення ефективності використання AR-технологій в ігрових додатках.

Вихідні дані для проведення роботи – аналіз існуючих AR-технологій та ігрових додатків, експериментальні вимірювання реакцій користувачів на взаємодію з AR-іграми, вивчення методів віртуальної реальності.

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Новизна запропонованих рішень полягатиме в досліджені та порівнняні методів взаємодії з Artificial Reality, що дозволить імплементувати їх в ігровий додаток та надати користувачам більш імерсивні відчуття від гри, роблячи її більш захоплюючою.

Практична цінність дослідження полягатиме в створенні високоякісних AR-ігор, які здатні не лише розважати користувачів, а й використовуватися у різних галузях, таких як освіта, медицина та бізнес. Це дозволить вдосконалити процеси навчання, покращити результати лікування за допомогою віртуальної реабілітації, а також забезпечить нові можливості для рекламодавців та бізнесу у сфері взаємодії з клієнтами через інноваційні AR-технології.

4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Результати досліджень мають бути подані у вигляді, що дозволяє оцінити переваги певних способів взаємодії з Artificial Reality над іншими. В результаті роботи повинен бути розроблений ігровий додаток, що включатиме в себе декілька способів взаємодії з AR середовищем.

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок – кінець**
Аналіз теми та постановка задачі	12.09.2022-30.09.2023
Дослідження способів та принципів взаємодії з Artificial Reality	01.10.2022-31.10.2023
Розробка ігрового додатку та імплементація в нього обраних на попередньому етапів способів взаємодії	01.11.2022-12.12.2023

6 РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект від реалізації результатів роботи полягає у зростанні продуктивності та зниженні витрат у виробництві AR-ігор, завдяки

знаходженню та вивченню способів взаємодії з Artificial Reality, що дозволяють швидше та ефективніше створювати ігрові додатки.. Це призведе до збільшення конкурентоспроможності компаній-розробників і, отже, до збільшення прибутків у цьому ринковому сегменті.

Соціальний ефект реалізації цих розробок виявиться у полегшенні доступу до високоякісних і захоплюючих AR-ігор для широкого кола користувачів. Зменшення витрат на розробку гри призведе до зниження її ціни для кінцевого споживача, зробить її доступнішою для людей з різних соціальних шарів. Крім того, спрощення процесу розробки і випуску AR-ігор зробить можливим швидше впровадження нових технологій у сфері розваг та освіти, що позитивно вплине на розвиток індустрії в цілому.

7 ДОДАТКОВІ ВИМОГИ

Додаткові вимоги не висуваються.

Завдання видав	_____ (підпис)	*доц. Приходченко С.Д* (прізвище, ініціали)
Завдання прийняв до виконання	_____ (підпис)	*Тріско А.Є.* (прізвище, ініціали)

Дата видачі завдання: *12.09.2023 р.*

Термін подання кваліфікаційної роботи до ЕК *12.12.2023*

РЕФЕРАТ

Пояснювальна записка: 96 стор., 34 рис., 4 додатка, 24 джерел.

Об’єкт дослідження: взаємодія з Artificial Reality (AR) у контексті ігрового додатку.

Предмет дослідження: методи та принципи взаємодії користувача з AR-середовищем в ігровому додатку.

Мета магістерської роботи: підвищення ефективності використання AR-технологій в ігрових додатках.

Методи дослідження: аналіз існуючих AR-технологій та ігрових додатків, психофізіологічні вимірювання реакцій користувачів на взаємодію з AR-іграми, розробка алгоритмів взаємодії та їх імплементація у вигляді прототипу гри.

Новизна отриманих результатів полягатиме в досліджені та порівнняні методів взаємодії з Artificial Reality, що дозволить імплементувати їх в ігровий додаток та надати користувачам більш імерсивні відчуття від гри, роблячи її більш захоплюючою.

Практична цінність результатів дослідження полягатиме в створенні високоякісних AR-ігор, які здатні не лише розважати користувачів, а й використовуватися у різних галузях, таких як освіта, медицина та бізнес. Це дозволить вдосконалити процеси навчання, покращити результати лікування за допомогою віртуальної реабілітації, а також забезпечить нові можливості для рекламодавців та бізнесу у сфері взаємодії з клієнтами через інноваційні AR-технології.

Область застосування. Розроблене програмне забезпечення може застосовуватися для аналізу взаємодії користувача з середовищем AR та для оцінки використання оптимальних технологій.

Значення роботи та висновки. Розроблене програмне забезпечення дозволяє взаємодіяти з Artificial Reality та аналізувати поведінку користувача під час цього.

Прогнози щодо розвитку досліджень. Дослідження та інтеграція додаткових способів та принципів взаємодії з AR.

Список ключових слів: Artificial Reality, ігровий додаток, ігровий двигун, користувач, Unity, Vuforia.

ABSTRACT

Explanatory note: 96 pages, 34 figures, 4 appendices, 24 sources.

The object of research: interaction with Artificial Reality (AR) within the context of a gaming application.

The subject of research:methods and principles of user interaction with the AR environment in a gaming application..

The purpose of the master's work: enhancing the efficiency of utilizing AR technologies in gaming applications.

Research methods: analysis of existing AR technologies and gaming applications, psychophysiological measurements of user reactions to interactions with AR games, development of interaction algorithms, and their implementation in the form of a game prototype.

The novelty of the obtained results will involve the exploration and comparison of interaction methods with Artificial Reality, enabling their implementation in the gaming application to provide users with a more immersive gaming experience, making it more captivating.

The practical value of the results lies in the creation of high-quality AR games capable of not only entertaining users but also being applied in various fields such as education, medicine, and business. This will refine the learning processes, enhance treatment outcomes through virtual rehabilitation, and introduce new opportunities for advertisers and businesses in customer interaction through innovative AR technologies.

Field of application.The developed software can be utilized to analyze user interaction within the AR environment and assess the utilization of optimal technologies.

Value of work and conclusions. The developed software enables interaction with Artificial Reality and allows the analysis of user behavior during this interaction.

Forecasts regarding the development of research. Further research will involve exploring and integrating additional methods and principles of interaction with AR.

Keyword list: Artificial Reality, gaming application, game engine, user, Unity, Vuforia.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AR – сукупність технологій, яка дозволяє об'єднувати реальний світ і віртуальний світ, додаючи до реальної картини або відео додаткові віртуальні об'єкти;

Unity – багатоплатформовий інструмент для розробки відеоігор і застосунків;

Vuforia – це комплект розробки програмного забезпечення доповненої реальності (SDK) для мобільних пристроїв, який дозволяє створювати додатки доповненої реальності;

APK – формат архівних виконуваних файлів-програм для Android;

ПЗ – програмне забезпечення.

ЗМІСТ

ВСТУП

Доповнена реальність (AR) є одним із провідних та швидкозростаючих імерсивних досвідів XXI століття. AR викликала революцію в різних сферах, включаючи сфери охорони здоров'я і медицини, навчання та освіту, туризм, дизайн, виробництво та інші схожі галузі, прийняття яких прискорило розвиток AR надзвичайним чином. Технології відстеження є будівельними блоками AR і створюють точку відліку для руху та створюють середовище, де віртуальні та реальні об'єкти відображаються разом. Для досягнення реального досвіду з розширеними об'єктами пропонується кілька технологій відстеження.

Доповнена реальність надає користувачам комбінований вид, накладаючи комп'ютерно-генерований віртуальний контент, такий як звук, графіка, текст або відео, на об'єкти реального світу. Основні компоненти процесу в AR - відстеження положення для розміщення віртуальних об'єктів у реальному середовищі та відображення віртуального контенту користувачеві. Процес відстеження в AR полягає в слідуванні за визначеним шаблоном у реальному світі за допомогою комп'ютера чи мобільного пристрою для правильного розміщення віртуальних об'єктів у реальному світі. Технології відображення використовуються для відображення віртуального контенту перед очима глядача.

Об’єкт досліджень: взаємодія з Artificial Reality (AR) у контексті ігрового додатку.

Предмет дослідження: методи та принципи взаємодії користувача з AR-середовищем в ігровому додатку.

Мета магістерської роботи: підвищення ефективності використання AR-технологій в ігрових додатках.

Методи дослідження: аналіз існуючих AR-технологій та ігрових додатків, психофізіологічні вимірювання реакцій користувачів на взаємодію з AR-іграми, розробка алгоритмів взаємодії та їх імплементація у вигляді прототипу гри.

Наукова новизна отриманих результатів полягатиме в дослідженні та порівнянні методів взаємодії з Artificial Reality, що дозволить імплементувати їх в ігровий додаток та надати користувачам більш імерсивні відчуття від гри, роблячи її більш захоплюючою.

Практичне значення результатів дослідження полягатиме в створенні високоякісних AR-ігор, які здатні не лише розважати користувачів, а й використовуватися у різних галузях, таких як освіта, медицина та бізнес. Це дозволить вдосконалити процеси навчання, покращити результати лікування за допомогою віртуальної реабілітації, а також забезпечить нові можливості для рекламодавців та бізнесу у сфері взаємодії з клієнтами через інноваційні AR-технології.

Структура та обсяг дипломної роботи: Кваліфікаційна робота складається з трьох розділів та висновків. Містить 93 сторінки тексту, 34 рисунка, 24 використаних джерела та 4 додатки.

РОЗДІЛ 1.
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ

1.1. Загальний опис предметної області та її поточний стан

З розвитком технологій доповненої реальності майбутнє обіцяє занурити нас у світ, де межа між реальністю та віртуальністю стане ще тоншою і менш помітною. Сучасні події в історії розвитку AR лише відкривають шлях до неймовірних перспектив, але майбутнє може приховувати неймовірні можливості, про які сьогодні важко уявити.

Щодня технології AR зазнають швидкого розвитку, і це відкриває безліч нових горизонтів. Невдовзі можна буде очікувати виходу на ринок більш досконалої апаратури, яка буде не лише компактнішою, але й надзвичайно потужною, інтегруючись непомітно в наше повсякденне життя. Від домашнього оточення до робочого простору та навіть вуличних ландшафтів, AR може стати неот'ємною частиною нашого середовища, обогащуючи реальність неймовірними можливостями, які ще не відомі.

Майбутнє AR виглядає захопливо та вражаюче, і обіцяє внести кардинальні зміни в те, як ми сприймаємо світ. Від ігор та розваг до освіти та виробництва - можливості застосування AR здаються безмежними. Однак, перш ніж ми віддамося фантазіям про майбутнє, важливо приглянутися до минулого та поточного етапу розвитку цієї технології.

Розглядаючи минулі досягнення та поточні тенденції, ми можемо краще зрозуміти, яким чином AR дійшла до свого сучасного стану. Вивчення цих векторів розвитку дозволяє краще оцінити та передбачити перспективи цієї технології в майбутньому. Важливо розуміти, які фактори та події визначили сучасний вигляд AR, і як ці впливи можуть визначати її майбутнє. Тільки осяяний минулим, ми зможемо повністю оцінити та віддати себе захоплюючим перспективам, які принесе AR у світі завтрашнього дня.

Рис 1.1 Приклад використання доповненої реальності вдома

У 1968 році американський вчений і піонер інтернету Іван Сазерленд винаходить гарнітуру для голови як вікно до віртуального світу.[1] Проте тодішні технології робили його винахід непрактичним для широкого використання.

Сазерлендова гарнітура взагалі може бути визнана піонерською, оскільки вона відкрила шлях до майбутнього розвитку віртуальної реальності. У той час, коли технічні обмеження значно утруднювали можливості пристрою, сам факт створення ідеї гарнітури для голови говорив про велику винахідливість Сазерленда та його віру в потенціал віртуальної технології.

Технологічний ландшафт 1968 року не був готовим для повноцінного впровадження віртуальної реальності в повсякденне життя. Обчислювальна потужність комп'ютерів того часу була обмеженою, а розробка графіки та інтерфейсів знаходилася в ранній стадії. Такі обставини ускладнювали створення повноцінного та ефективного пристрою для віртуальної реальності.

Незважаючи на технічні труднощі, пристрій Сазерленда виявився важливим кроком для розвитку віртуальної реальності. Його гарнітура стала відомим початком для подальших досліджень та інновацій в цьому напрямку

Рис 1.2 "Меч Дамокла" - перша AR гарнітура

У 1975 році американський комп'ютерний художник Майрон Крюгер розробив перший інтерфейс "віртуальної реальності" у формі "Videoplace", який дозволяв користувачам маніпулювати та взаємодіяти з віртуальними об'єктами в реальному часі.

"Videoplace" став першим кроком у напрямку створення іммерсивних віртуальних середовищ, де користувачі могли не лише спостерігати за віртуальними об'єктами, але й взаємодіяти з ними, відчуваючи власну активну участь у цьому цифровому світі. Крюгер використовував датчики руху та відеокамери для реалізації інтерактивності, що створювало враження повноцінного взаємодії з віртуальним простором.

Його розробка визначила новий етап у розвитку віртуальної реальності, де артистична креативність поєднувалася з передовими технічними здобутками. "Videoplace" дало поштовх для подальших досліджень та експериментів в галузі віртуальної технології. Його внесок підкреслив, що віртуальна реальність не обмежується лише інженерними аспектами, але також стає сценою для прояву можливостей людської фантазії.

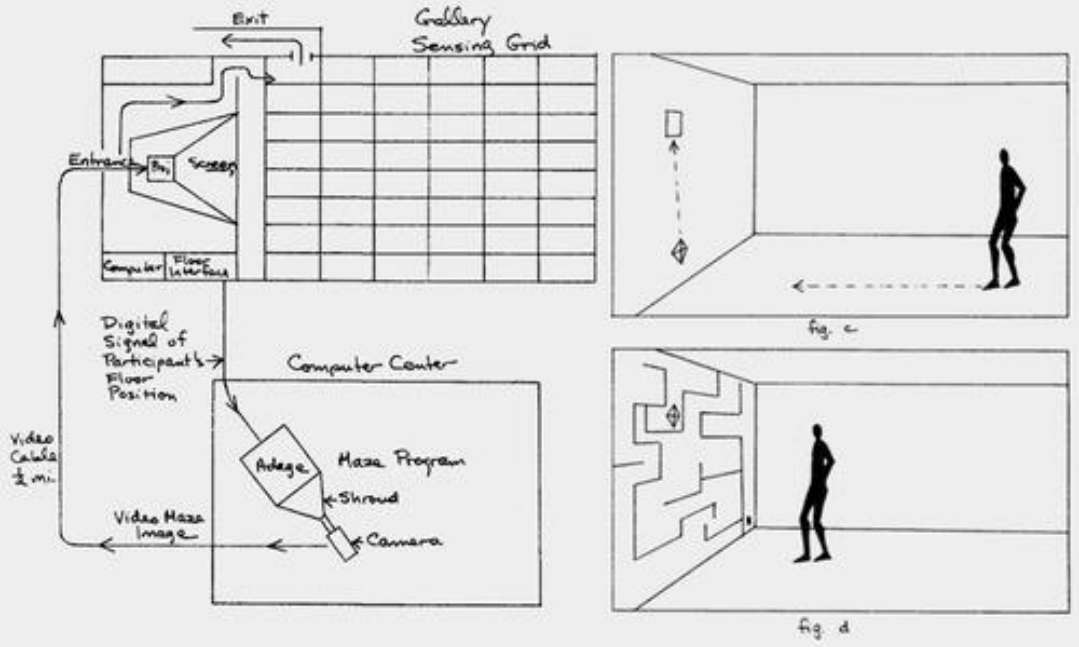

Рис 1.3 Схема роботи "Videoplace"

У 1980 році дослідник у галузі обчислювальної фотографії Стів Манн представив світові носимі обчислювальні пристрої.

Рис 1.4 Стів Манн та його переносний комп’ютер

Тоді, звісно, ще не існували терміни "віртуальна реальність" чи "доповнена реальність". Термін "віртуальна реальність" був вигаданий Джароном Лейнером у 1989 році, а фразу "доповнена реальність" вперше вжив Томас Коделл з Boeing у 1990 році.

Найімовірніше, перша дійсно працююча система доповненої реальності була розроблена Луїсом Розенбергом у 1992 році в Лабораторії досліджень Армстронга Повітряних сил США. Ця система називалася "Virtual Fixtures" і була надзвичайно складною робототехнічною системою, створеною для компенсації відсутності потужності високошвидкісної обробки 3D-графіки на початку 90-х років. Вона дозволяла показувати важливу інформацію над робочим середовищем для підвищення продуктивності людини.[2]

Рис 1.5 Луїс Розенберг в системі "Virtual Fixtures"

В 1997 Рональд Азума сформував визначення AR-технології[3] як таку, що повинна відповідати трьом основним вимогам:

– об'єднувати віртуальний та реальний контент;
– бути інтерактивною в реальному часі;
– мати реєстрацію в тривимірному просторі.

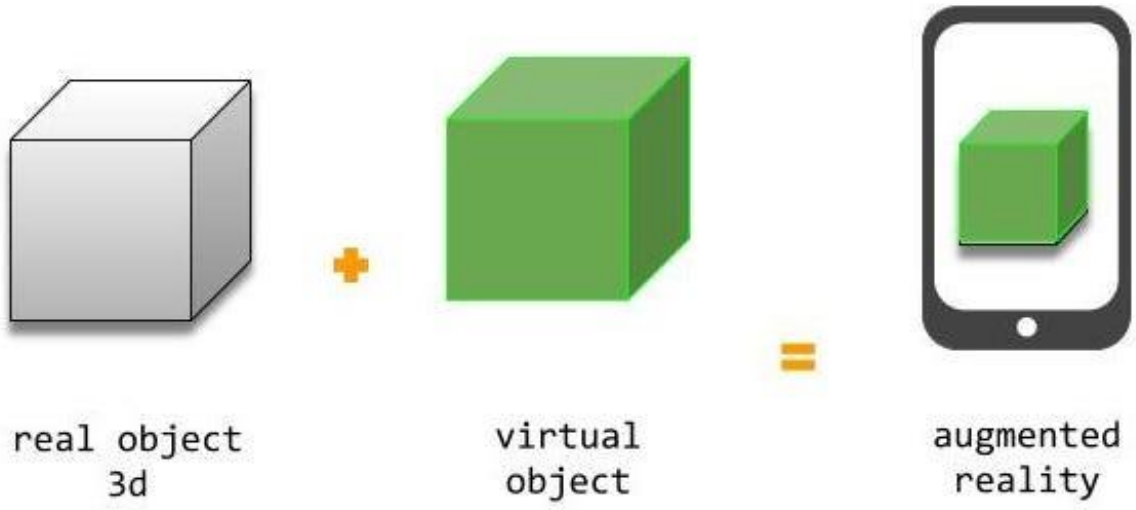

Рис 1.6 Складові аугументованої реальності

Доповнена реальність пройшла вражаючий шлях від скромних прототипів до надзвичайно поширеної технології, якою вона є сьогодні. Її вплив на різні галузі суспільства надзвичайно значущий, а застосування AR продовжують розширюватися, переносячи нас в новий етап технологічного розвитку.

Один із очевидних проявів успішного впровадження AR - віртуальні вітрини та візуалізатори в інтернет-магазинах. Користувачі тепер можуть випробовувати вироби чи переглядати товари перед покупкою, отримуючи відмінне споживче враження та покращуючи якість онлайн-шопінгу. Це лише один із прикладів того, як AR змінює спосіб, яким ми сприймаємо та взаємодіємо з віртуальним світом.

Застосування AR також виявилося надзвичайно корисним у медичній сфері. Технології, такі як AccuVein, дозволяють медичним працівникам швидко та точно локалізувати вени пацієнта, що полегшує проведення процедур та зменшує стрес для пацієнтів. Такі інновації сприяють покращенню ефективності медичних послуг та рятуванню часу.

Зростанням можливостей доповненої реальності відкриваються нові перспективи, які перевершують очікування піонерів цієї технології. Від освіти та тренувань до розваг та виробництва, AR інтегрується в наше повсякденне життя,

перетворюючи його та надаючи нам можливість бачити світ навколо себе з абсолютно нової перспективи.

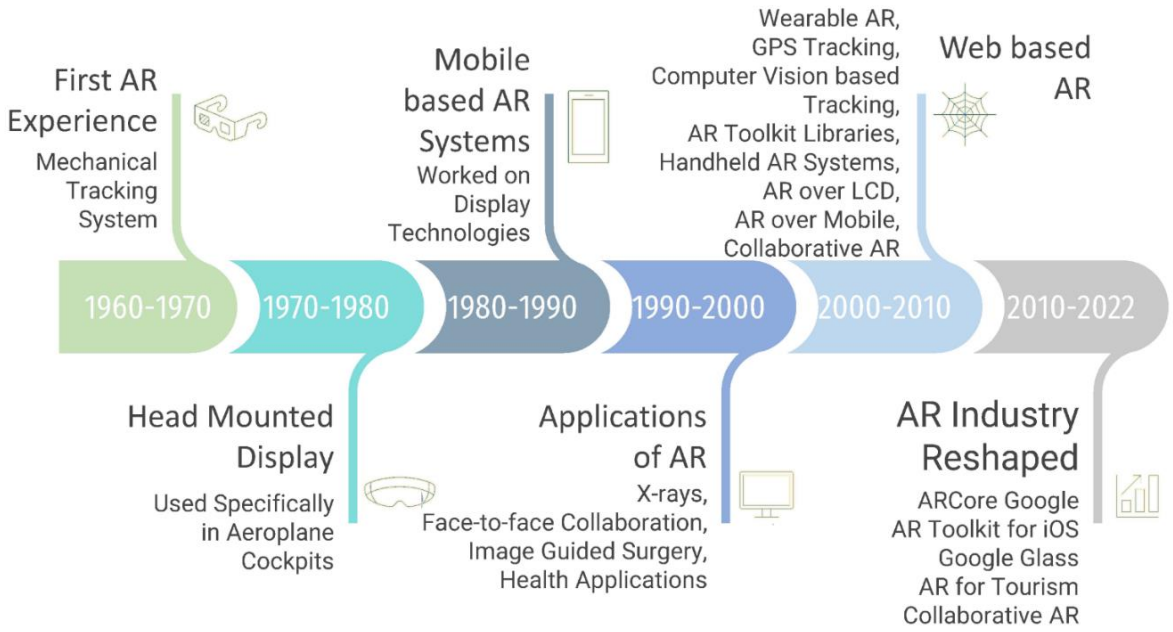

Рис 1.7 Етапи розвитку AR технології

1.2. Огляд принципів та технологій взаємодії з Artificial Reality

Технології відстеження грають важливу роль у впровадженні розширеної реальності, забезпечуючи користувачам можливість взаємодії з віртуальним середовищем та оточуючим світом. Імплементація конкретних технологій відстеження залежить від різних факторів, таких як тип середовища, вид даних, які необхідно відстежувати, та фінансові можливості.

Однією з ключових властивостей технологій відстеження є їх здатність забезпечувати відчуття руху віртуального об'єкта або користувача у розширеному середовищі. Це робить можливим переміщення віртуальної реальності та розширеного середовища, створюючи враження присутності та іммерсії.

Важливим аспектом є правильний вибір і встановлення системи відстеження. Це може включати в себе використання датчиків руху, камер, лазерів чи інших технологій для точного визначення положення та рухів об'єктів чи користувачів. Найбільш ефективний вибір технології залежить від контексту

використання, такого як віртуальні ігри, навчання, медицинські додатки чи промислові застосування.

Розгляд та порівняння основних типів технологій відстеження, їхніх призначень, переваг та недоліків є важливим етапом в розумінні, які саме аспекти були враховані при імплементації конкретних технологій у конкретних розробках. Такий аналіз допомагає обрати оптимальні рішення, враховуючи конкретні потреби та обмеження кожного випадку використання.

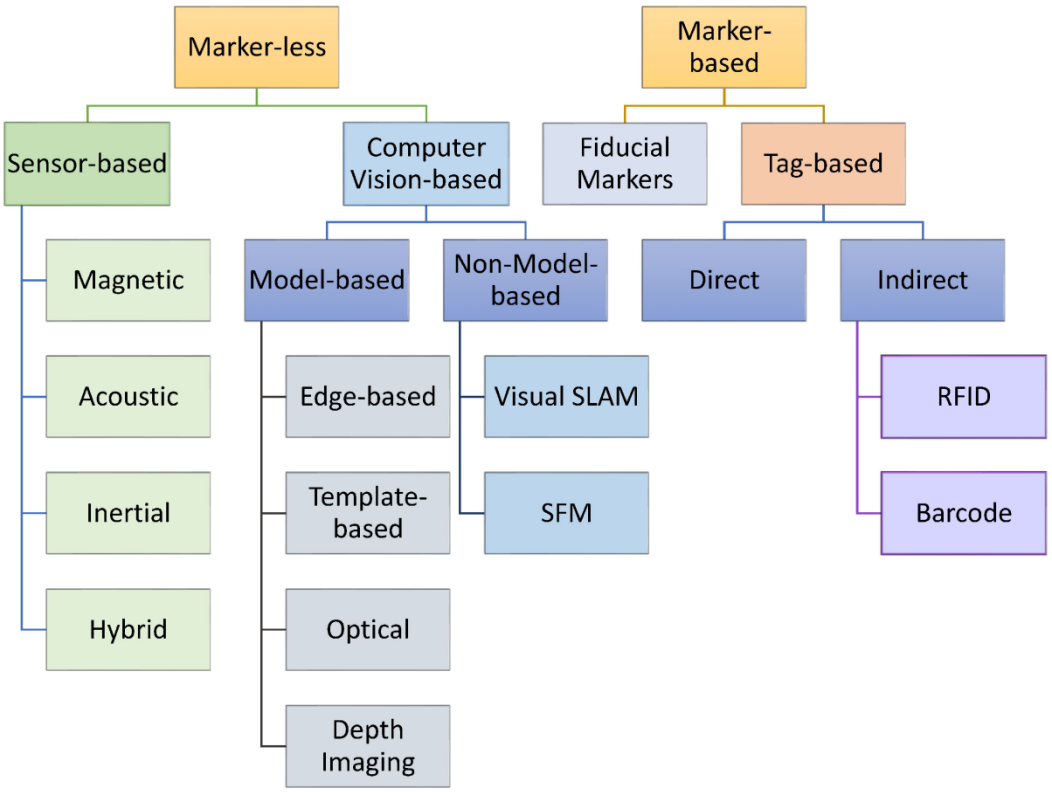

Рис 1.8 Категоризація технологій відстеження

Технологія магнітного відстеження: ця технологія включає в себе джерело відстеження та два сенсори: один для голови та інший для руки. Джерело відстеження створює електромагнітне поле, в якому розташовані сенсори. Потім комп'ютер розраховує орієнтацію та положення сенсорів на основі затухання сигналу у полі. Це надає можливість повного діапазону рухів у 360 градусів, тобто дозволяє нам дивитися та рухатися у всіх трьох осях ординат.[4]

Однією з основних проблем магнітного відстеження є обмежений діапазон розпізнавання. Орієнтацію та положення можна визначити, налаштовуючи

приймач відносно спостерігача. Однак роздільна здатність магнітного поля зменшується з четвертою ступеню відстані, а сила магнітного поля зменшується з кубом відстані. Тому магнітні відстежувачі мають обмежений робочий об'єм. Крім того, магнітні відстежувачі чутливі до оточуючих середовища магнітних полів і типу використаного магнітного матеріалу, а також схильні до відхилень вимірювання.

Незважаючи на ці обмеження, магнітна технологія відстеження активно використовується в різних галузях, включаючи обслуговування, медицину та виробництво. У сфері обслуговування вона може забезпечувати точне відстеження рухів користувача, а в медицині використовується для навігації та взаємодії з віртуальними об'єктами під час хірургічних втручань. У виробництві ця технологія може поліпшити точність та ефективність роботи операторів під час виконання завдань у віртуальному середовищі.

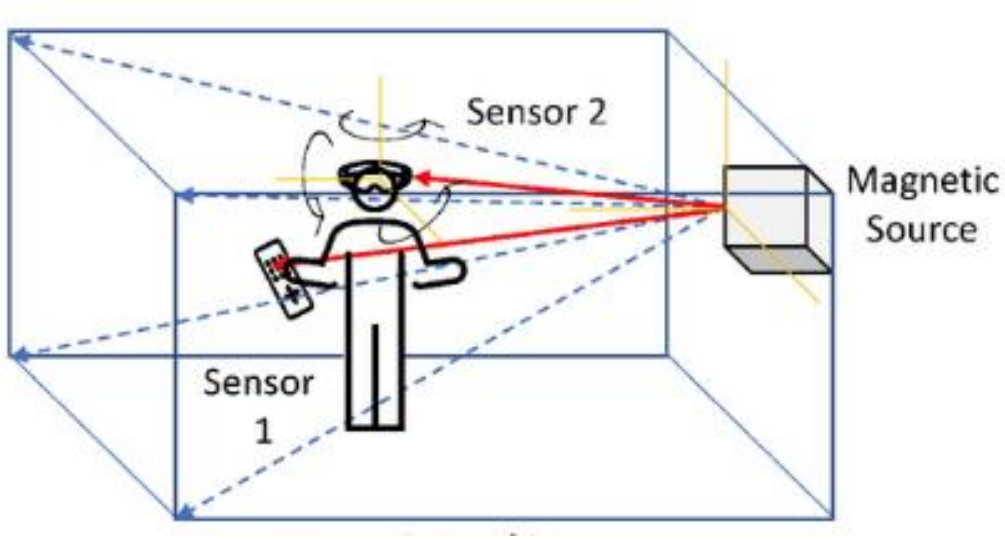

Рис 1.9 Схема роботи магнітного стеження

Відстеження за допомогою інерційних сенсорів: магнітометри, акселерометри та гіроскопи є прикладами інерційних вимірювальних блоків (ІВБ), які використовуються в інерційному відстеженні для оцінки швидкості та орієнтації відстежуваного об'єкта. Інерційна система відстеження використовується для визначення трьох ротаційних ступенів свободи відносно гравітації. Крім того, можна визначити період часу оновлення трекерів та інерційну швидкість за зміною положення трекера.

Переваги інерційного відстеження: воно не потребує прямого видимості та не має обмежень дальності. Воно не схильне до оптичних, акустичних, магнітних та електромагнітних джерел перешкод. Крім того, воно має незначний час затримки і може бути оброблене так швидко, як це потрібно.

Недоліки інерційного відстеження: воно схильне до дрейфу орієнтації та положення з часом, основний вплив має на вимірювання положення.[5]

Дрейф може виникнути через накопичення помилок вимірювань, що призводить до неточностей у визначенні положення об'єкта в просторі. Для компенсації цього ефекту, інерційні системи можуть використовувати інші технології відстеження або методи корекції даних для покращення точності результатів.

Усупереч цим обмеженням, інерційне відстеження залишається важливим інструментом в багатьох застосунках, таких як віртуальна реальність, аерокосмічна індустрія, спорт, та інші, де точність та швидкість реакції мають велике значення

Візійне відстеження визначається як метод відстеження, що встановлює положення камери за допомогою оптичних сенсорів. Цей підхід стає все більш популярним у сфері розширеної реальності, завдяки покращеній обчислювальній потужності споживчих пристроїв та загальній поширеності мобільних пристроїв, таких як планшети та смартфони, які роблять їх доступнішими для впровадження технологій AR.

Основна перевага візійного відстеження полягає у здатності точно визначати положення камери та відстежувати рух об'єктів у реальному часі. Це важливо для створення іммерсивного AR-досвіду, оскільки система може реагувати на рухи користувача або оточуючих об'єктів. Крім того, розвиток оптичних сенсорів та алгоритмів обробки зображень дозволяє покращити точність та стабільність візійного відстеження.

Основними недолікомами візійного відстеження є знижена чіткість відстеження для близьких об'єктів і залежність від обмежень технічних

характеристик пристрою користувача, таких як якість камери та обчислювальна потужність.

Відстеження тривимірної структури: для визначення тривимірних точок у сцені можуть використовуватися різні типи сенсорів. Найбільш поширеними є структуроване світло або принцип часу польоту.[6] Ці технології працюють на принципі аналізу глибини. У цьому випадку глибинна інформація про реальне оточення витягається за допомогою розмітки та відстеження.

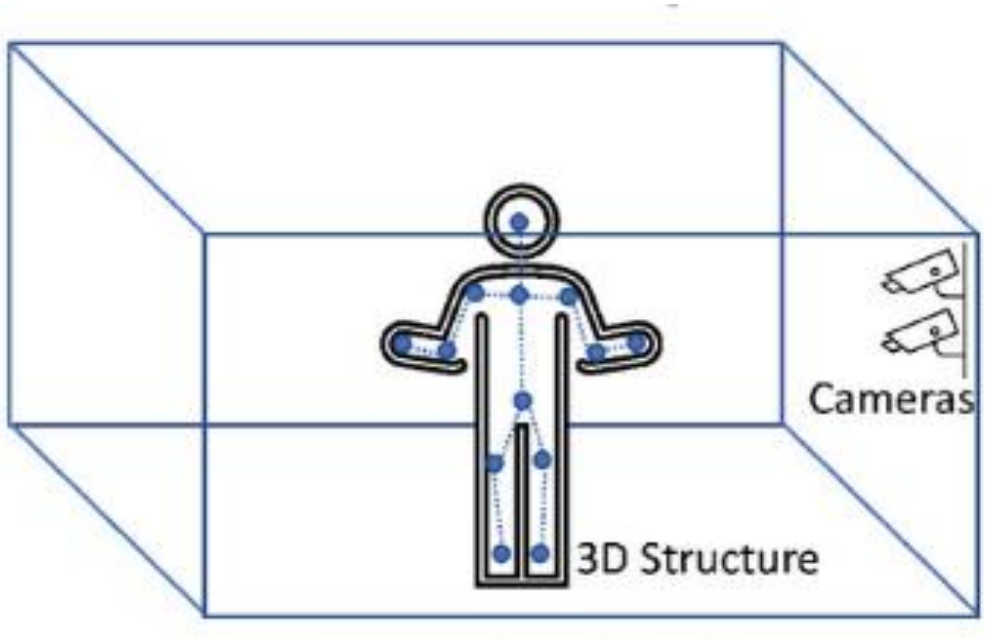

Рис 1.10 Схема роботи відстеження тривимірної структури

Розвиток комерційних сенсорів здатних до виконання вимірювань глибини дозволяє легше і доступніше впровадження технологій відстеження тривимірної структури в сучасних системах розширеної реальності. Це дозволяє створювати більш реалістичні та точні AR-досвіди для користувачів.

Інфрачервоне відстеження: об'єкти, які випромінюють або відбивають світло, є одними з найраніших методів відстеження, які використовуються в технологіях розширеної реальності. Їх висока яскравість порівняно з навколишнім середовищем робить це дуже простим.[7] Самостійно випромінюючі мішені також були байдужі до навколишнього середовища, наприклад, тіні або поганого освітлення. Крім того, ці мішені можуть бути або закріплені на відстежуваному об'єкті та камері ззовні об'єкта і відомі як

"зовнішній погляд". Або це може бути "внутрішній погляд", зовнішній у середовищі з камерою, прикріпленою до мішені. Внутрішня конфігурація, порівняно із зовнішньою системою, має більшу роздільну здатність та вищу точність кутової орієнтації. Внутрішня конфігурація використовується при розробці кількох систем, зазвичай з інфрачервоними світлодіодами, розміщеними на стелі, та гарнітурою з камерою, що дивиться ззовні.

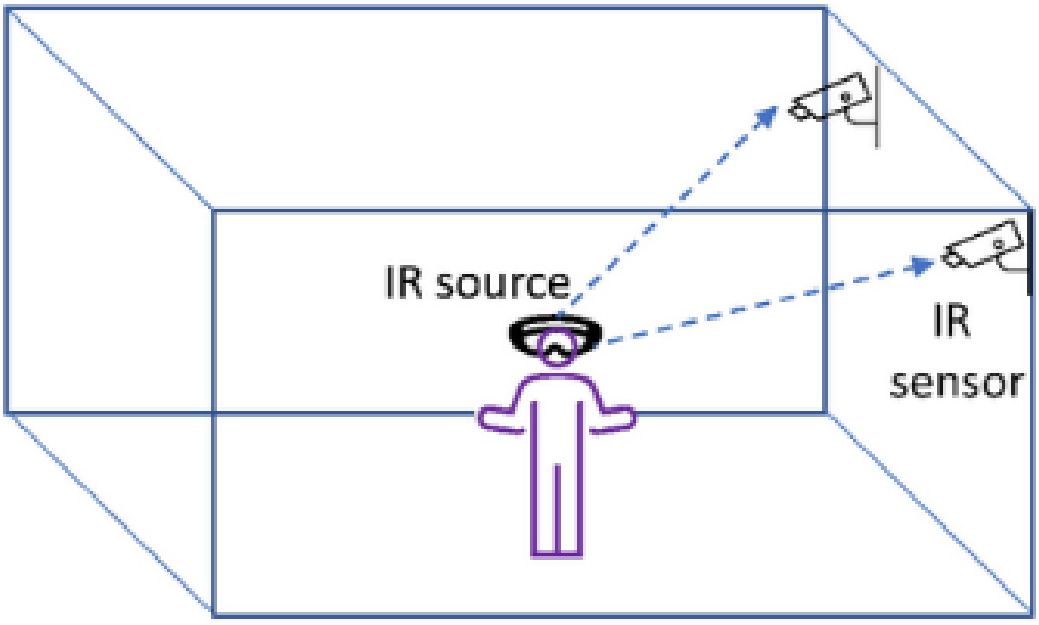

Рис 1.11 Схема роботи інфрачервоного відстеження

Відстеження на основі моделі: раніше відстеження тривимірної моделі об'єкта зазвичай створювалося вручну. У цій системі, для вилучення структурної інформації з сцени використовувалися крайові фільтри, які розпізнавали контури об'єктів.[8] Для визначення позиції об'єктів порівнювалася інформація про краї та геометричні примітиви. Цей підхід дозволяв досягти стабільного відстеження об'єктів навіть у відкритих середовищах, де може бути значна різноманітність об'єктів та фонів.

Однак, із зростанням складності сцен та вимог до точності відстеження, з'явилася необхідність розробки більш вдосконалених алгоритмів стеження та більш складних моделей об'єктів. Використання технологій машинного навчання, комп'ютерного зору та глибокого навчання дозволяє автоматизувати

процес створення та використання тривимірних моделей для ефективного відстеження об'єктів у реальному часі.

Подібний підхід дозволяє досягти стабільного відстеження об’єктів навіть у відкритих середовищах. Мінусами є необхідність розробки алгоритмів стеження та моделей об’єктів.

GPS-відстеження: ця технологія відноситься до позиціонування за допомогою технології GPS (глобальна система позиціонування), що використовує супутникові сигнали для визначення точної позиції об'єкта на поверхні Землі.

На сьогоднішній день точність GPS-систем може сягати до 3 метрів, але завдяки прогресу в супутникових технологіях і різних розробках, можливе покращення цієї точності. Один із варіантів покращення - це використання реального кінематичного методу (RTS), який базується на несучій хвилі сигналу GPS. Його основна перевага полягає в тому, що він може підвищити точність до рівня сантиметрів, що робить його особливо корисним у вимірах великої точності.

GPS-відстеження широко використовується в різних областях, включаючи військові застосування, ігрову індустрію[9] та додатки для проведення екскурсій у доповненій реальності. Однак, через обмежену точність відстеження позиції, його використовують у гібридних системах відстеження або в тих випадках, де важлива не сама точна реєстрація позиції, але загальне положення об'єктів у просторі.

Рис 1.12 Схема роботи GPS-відстеження

Гібридне відстеження є передовим напрямком в розробці систем позиціонування, призначених для досягнення визначених цілей, які включають в себе покращення точності, усунення вад, що виникають у використанні окремих систем, а також надання більшої свободи руху та огляду для користувачів. Одним із прикладів гібридного відстеження є поєднання GPS і візійного відстеження, що дозволяє компенсувати неточності позиціонування, характерні для GPS.

Головною перевагою гібридного підходу є можливість зниження впливу слабкостей окремих технологій та покращення точності визначення положення. Наприклад, там, де GPS може бути вразливим до перешкод або мати обмежену точність, візійне відстеження може забезпечити додаткову інформацію для уточнення місця розташування об'єктів[10]. Такий підхід особливо корисний у ситуаціях, де важлива велика точність позиційного визначення.

Звісно, гібридне відстеження також має свої виклики. Наприклад, потреба в інтеграції та синхронізації декількох технологій в одній системі може бути трудомісткою. Однак, не зважаючи на ці труднощі, гібридні системи відстеження вже сьогодні використовуються в різних галузях, надаючи користувачам надійні та точні дані про положення в реальному часі.

Fiducial Tracking, або відстеження фідуціалів, визначається використанням штучних орієнтирів, відомих як фідуціали, які вбудовуються в середовище для покращення процесу відстеження та реєстрації об'єктів. Складність використання фідуціалів залежить від конкретної технології та застосування.

У ранніх системах фідуціали можуть бути реалізовані у вигляді аркушів паперу або невеликих кольорових світлодіодів. Їх виявлення відбувалося за допомогою відповідності кольорів або інших характеристик, і ці фідуціали додавались до оточуючого середовища. Якщо системі відомі позиції фідуціалів, та їх достатньо для виявлення на сцені, можна визначити положення камери або іншого спостерігача.

Важливою характеристикою фідуціального відстеження є можливість динамічного розширення робочого простору. Знаючи позицію одного або кількох фідуціалів, можна додавати нові фідуціали, розширюючи область відстеження. З розвитком цієї концепції виникали більш складні функціональності, такі як багаторядкове розпізнавання фідуціалів на великих відстанях.

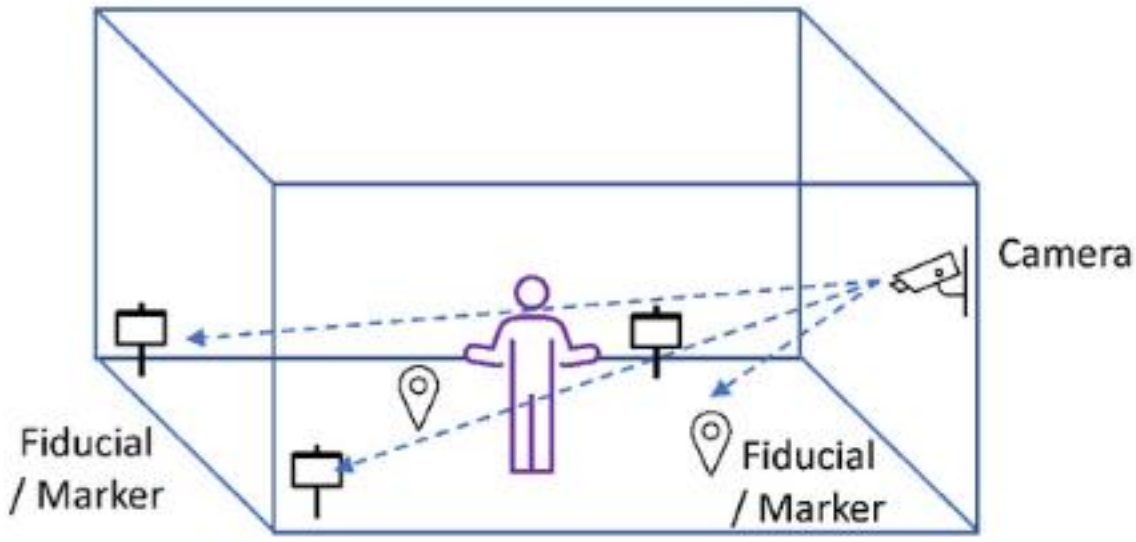

Рис 1.13 Схема роботи фідуційного відстеження

Для визначення точного положення спостерігача за допомогою фідуціалів, необхідно мати інформацію про мінімум чотири точки з відомою позицією. Це

може включати в себе позначення або розміщення фідуціалів, які вже визначені в просторі.[11]

1.3. Процес відстеження об’єкту в AR

Яка б технологія відстеження не була обрана в основу системи доданої реальності основні етапи її роботи завжди ідентичні і можуть бути зведені до пунктів, що описані нижче.

Виявлення об'єктів. Початковий етап відстеження об'єктів, на якому AR-система активно розпізнає та локалізує предмети інтересу в реальному середовищі користувача. Це може бути досягнуто різними методами, в залежності від технології, використовуваної в системі. Наприклад, це може включати в себе використання візійного відстеження, сенсорів руху, інфрачервоного відстеження або GPS.[12]

Розпізнавання об'єктів. Після виявлення об'єктів AR-система порівнює їх з базою даних відомих об'єктів для їхнього розпізнавання та категоризації. Для цього використовуються техніки розпізнавання об'єктів, такі як порівняння зображень, витягування ознак або навчання машин. Цей етап дозволяє системі ідентифікувати реальні об'єкти та взаємодіяти з ними, пов'язуючи їх з віртуальною інформацією.[13]

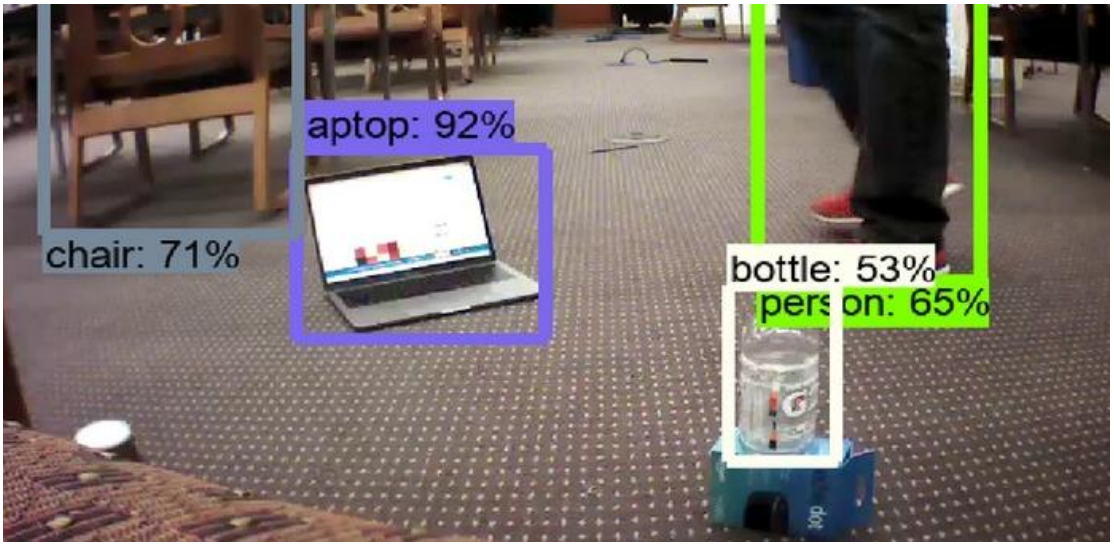

Рис 1.14 Приклад розпізнавання об’єкта

Відстеження об'єктів та визначення положення. пЯк тільки об'єкти розпізнані, AR-система надає можливість відстежувати їхні рухи та визначає їхнє положення (позицію та орієнтацію) в реальному часі. Це досягається за допомогою постійного аналізу даних від різних датчиків, таких як камери, гіроскопи, та акселерометри, що дозволяє оновлювати інформацію про положення та орієнтацію віртуального контенту щодо відстежуваних об'єктів.[14]

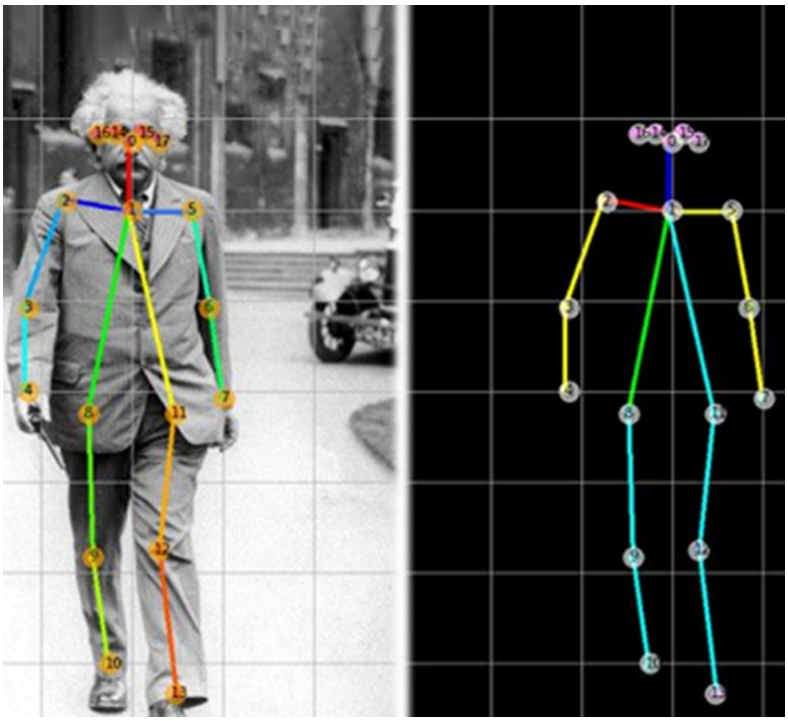

Рис 1.15 Приклад визначення положення людини

Прив’язка віртуального контенту. Після того як об'єкти відстежуються та їхня позиція оцінюється, AR-система прив'язує віртуальний контент до фізичних об'єктів. Це передбачає вирівнювання віртуальних елементів з відстежуваними об'єктами в реальному часі так, щоб вони здавалися частиною реального оточення та взаємодіяли з ним без проблем. До віртуального контенту можна включати 3D-моделі, фотографії, відео та інші цифрові матеріали, які покращують враження користувача від навколишнього середовища.[15]

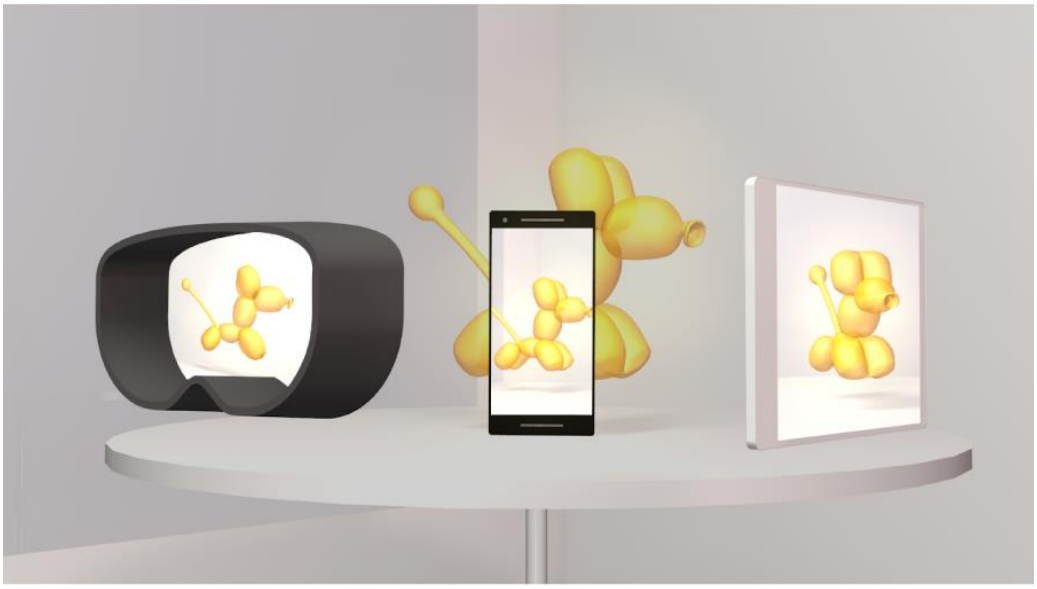

Рис 1.16 Приклад прив’язки об’єкта

Робота з оклюзією. Оклюзія, або вміння системи працювати з перекриттям об'єктів, є важливою складовою AR-технології. AR-система повинна забезпечити віртуальний контент таким чином, щоб враховувати оклюзію між реальними та віртуальними об'єктами. Наприклад, коли віртуальний об'єкт розташований за реальним об'єктом, система має забезпечити, що віртуальний об'єкт буде відображено як перекритий ближчим об'єктом, створюючи більш реалістичний та захоплюючий досвід використання системи[16]

Рис 1.17 Порівняння кадру без та з оклюзією

1.4 Висновки з першого розділу

В першому розділі мною було розглянута історія та поточний стан сфери Artificial Reality, її основні характеристики та вимоги. Також мною були проаналізовані існуючі технології та принципи взаємодії з AR, описані їх переваги та недоліки. Для подальшої роботи будуть використовуватись фідуційний та візійні підходи як такі, що є найбільш доступними та зручними для сумісного використання, адже саме завдяки цьому тандему я зможу отримати найбільш оптимальний результат в рамках поточної кваліфікаційної роботи.

РОЗДІЛ 2.
ОПИС МЕТОДІВ ВИРІШЕННЯ ЗАДАЧІ

2.1. Опис математичних методів в системах Augmented Reality

Системи доповненої реальності (AR) використовують різноманітні математичні методи для досягнення точності та ефективності у виявленні, відстеженні та взаємодії об'єктів у реальному часі. Ось деякі з них:

Геометричні методи: Включають геометричні обчислення для визначення відстані, кута та позиції об'єктів у просторі. Ці методи використовуються для визначення відстані між камерою та об'єктом, а також для розрахунку їхньої орієнтації.

Комп'ютерний зір: Використовує алгоритми обробки зображень та комп'ютерного зору для виявлення та розпізнавання об'єктів у реальному часі. Це включає в себе методи витягування ознак, співставлення зображень та визначення контурів. Одним з таких методів є триангуляція.

Триангуляція - це процес визначення точки в 3D-просторі та її проекції на задані зображення. 3D-моделювання з відео - це процес, де дані отримуються з відео, після чого ці дані перетворюються в хмару точок, яке поділяється на регіони перед проекцією. Послідовність зображень об'єкта - це серія проекцій поверхні цього об'єкта в 2D-простір. Кожне зображення залежить від камери, розташованої в тій точці огляду, яка, в свою чергу, залежить від відповідної матриці проекції камери.[17]

Метод триангуляції можна описати у термінах функції $\mathbf{x} \sim \tau(\mathbf{y}'_1, \mathbf{y}'_2, \mathbf{C}_1, \mathbf{C}_2)$, де y1′y1′ і y2′y2′ - це однорідні координати виявлених точок на зображенні, а C1C1 і C2C2 - це матриці камери. x (3D-точка) - це однорідне представлення отриманої 3D-точки. Знак ∼ позначає, що функція ττ повинна лише виробляти вектор, який дорівнює x з точністю до множення на ненульовий скаляр, оскільки в даному випадку використовуються однорідні вектори.

Фільтр Калмана: Використовується для покращення точності відстеження об'єктів, особливо при роботі з даними сенсорів, які мають шум чи неперіодичні відхилення. Маючи модель системи, фільтр Калмана може передбачати, яким буде стан системи у наступний момент часу. Саме це дозволяє фільтру настільки ефективно видаляти шум та оцінювати параметри, які не спостерігаються (не вимірюються) безпосередньо.[18]

Фільтр Калмана накладає обмеження на використовувані моделі: вони повинні бути дискретними моделями у просторі станів. Крім того, вони повинні бути лінійними. Цю модель можна зручно представити у вигляді різницевого матричного рівняння: $x_k = Fx_{k-1} + Bu_k + w_k$

Методи віртуальної реальності: AR може використовувати математичні моделі для створення віртуальних об'єктів у реальному часі, включаючи 3D-моделі, текстури, освітлення та рендерінг.

Методи глибинного навчання: Використовують глибинні нейронні мережі для аналізу глибини сцени та визначення розташування об'єктів у тривимірному просторі.

Для виявлення об'єктів широко використовується тип глибокої нейронної мережі, що називається згортковою нейронною мережею (CNN). CNN - це відомий алгоритм глибокого навчання, який використовує алгоритм зворотного поширення помилок у нейромережі з колекцією нейронів, впорядкованих у ієрархічних шарах. Вона також має типові характеристики нейронних мереж, такі як кілька зв'язаних між собою прихованих шарів. CNN для виявлення об'єктів навчається на великих позначених наборах даних та архітектурах нейромережі, які вивчають характеристики безпосередньо з даних без явної інженерії ознак.[19]

Ці математичні методи і їх комбінації допомагають системам доповненої реальності створювати реалістичні та імерсивні взаємодійні враження для користувачів.

2.2. Опис використаної архітектури та шаблонів проектування

В якості основи для розробленого ігрового додатку було взято ігровий двигун Unity. Сам двигун використовує і підштовхує своїх користувачів до імплементації архітектури на основі компонентів (Component-based architecture).

Архітектура на основі компонентів фокусується на розкладанні дизайну на окремі функціональні або логічні компоненти, які представляють собою чітко визначені інтерфейси зв'язку із методами, подіями та властивостями. Вона забезпечує вищий рівень абстракції і розбиває проблему на підпроблеми, кожна з яких пов'язана з частинами компонентів.

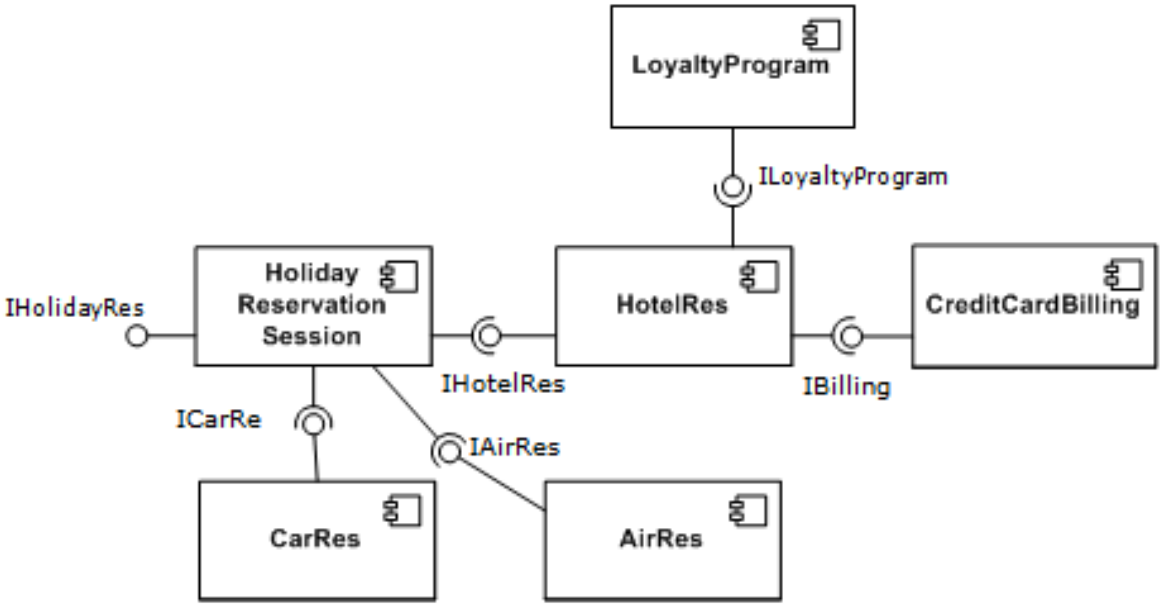

Рис 2.1 Приклад системи на основі компонентів

Основною метою архітектури на основі компонентів є забезпечення можливості повторного використання компонентів. Компонент інкапсулює функціональність та поведінку елемента програмного забезпечення в перевикористаний та автономний бінарний блок. Компонентно-орієнтований

дизайн програмного забезпечення має багато переваг порівняно з традиційними об'єктно-орієнтованими підходами, такими як:

- Зменшення часу введення на ринок та витрат на розробку за рахунок повторного використання існуючих компонентів.
- Підвищення надійності за рахунок використання існуючих компонентів у повторних проектах.

Що таке компонент? Компонент - це модульний, переносний, замінюваний та повторно використовуваний набір чітко визначеної функціональності, який інкапсулює свою реалізацію та використовує її як інтерфейс вищого рівня.

Компонент - це програмний об'єкт, призначений для взаємодії з іншими компонентами, який інкапсулює певну функціональність чи набір функціональностей. Він має чітко визначений інтерфейс і відповідає рекомендованому поведінці, яка спільна для всіх компонентів у межах архітектури.

Програмний компонент може бути визначений як одиниця композиції із контрактно визначеним інтерфейсом та явними залежностями від контексту. Це означає, що програмний компонент може бути незалежно розгорнутий і підлягає композиції третіми сторонами.

Характеристики компонентів:

- Повторне використання - компоненти зазвичай проектуються для повторного використання в різних ситуаціях та різних програмах. Однак деякі компоненти можуть бути призначені для конкретної задачі.
- Замінюваність - компоненти можуть бути вільно заміщені іншими схожими компонентами.
- Не залежать від контексту - компоненти призначені для роботи в різних середовищах і контекстах.
- Розширюваність - компонент може бути розширений за допомогою існуючих компонентів для надання нової функціональності.

- Інкапсуляція - компонент відображає інтерфейси, які дозволяють користувачу використовувати його функціональність, і не розкриває деталей внутрішніх процесів, внутрішніх змінних чи стану.
- Незалежність - компоненти проектуються з мінімальними залежностями від інших компонентів.

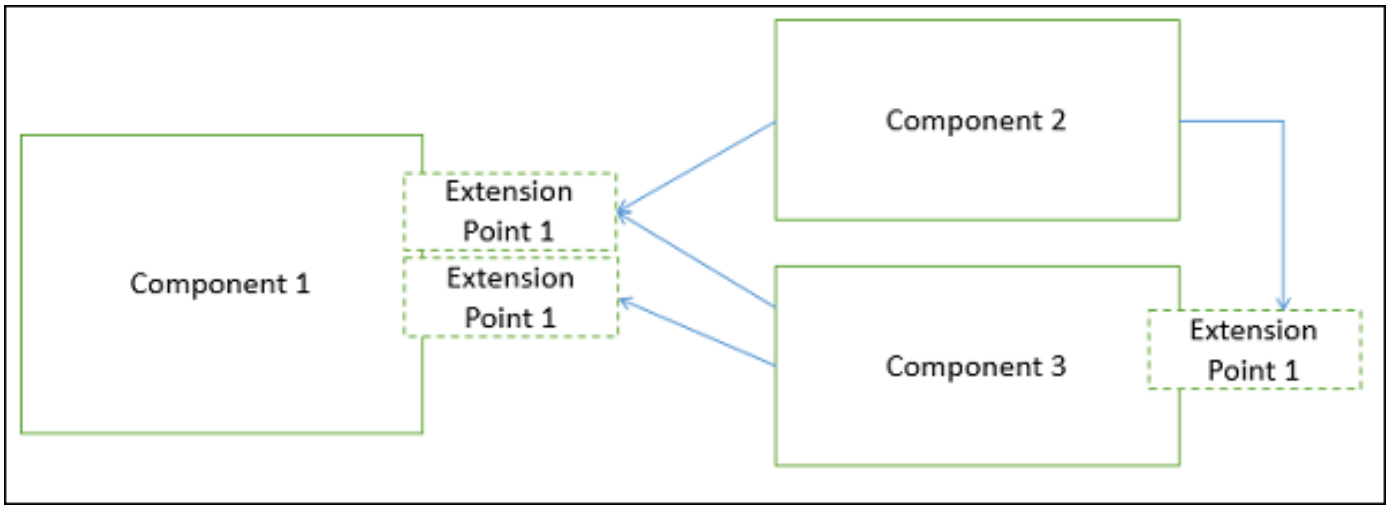

Рис 2.2 Схема взаємодії компонентів

Далі будуть описані шаблони проектування, що були використані протягом розробки проекту, їх плюси та мінуси.

Шаблон «Спостерігач» (Observer) - це шаблон проектування, який вирішує проблему багаторазового сповіщення про зміни стану одного об'єкта. Наприклад, коли об’єкт товару підписується на об’єкт гаманця, щоб знати коли грошей буде достатньо для покупки.[20]

Переваги:

- Дозволяє легко підписуватись та відписуватись від сповіщень за потреби
- Нема прямої залежності міжпідписниками та відсилаючим класом.
- Слідує принципу відкритості/закритості

Недоліки:

- Потрібно управляти підписниками суб'єкта, якщо вони повинні вийти з гри або бути видалені.
- У чистій реалізації немає можливості керувати порядком сповіщення підписників.

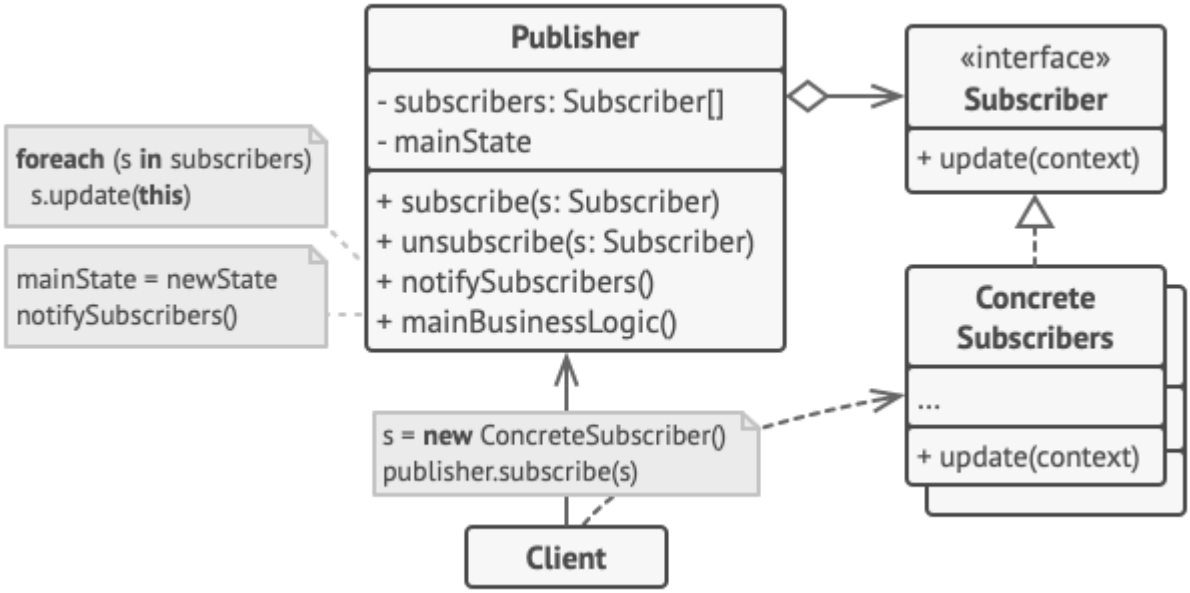

Рис 2.3 Схема взаємодії компонентів шаблона “Спостерігач”

Шаблон «Стратегія» (Strategy) - це шаблон проектування, який вирішує проблему заміни поведінки класу в залежності від вхідних даних або типу клієнта. Наприклад, коли прокладання маршруту на навігаторі залежить від наявності та типу вашого транспортного засобу.[21]

Переваги:

- Заміна алгоритму програми на льоту
- Ізолює код і данні алгоритмів від інших класів
- Слідує принципу відкритості/закритості

Недоліки:

- Потрібно мати конкретні критерії вибору стратегії.

- При розвитку додатка потрібна постійна підтримка коду, виділення загальних методів у базовий клас та винос частинних методів у клас, який їх використовує. Структура успадкування може розростатися.

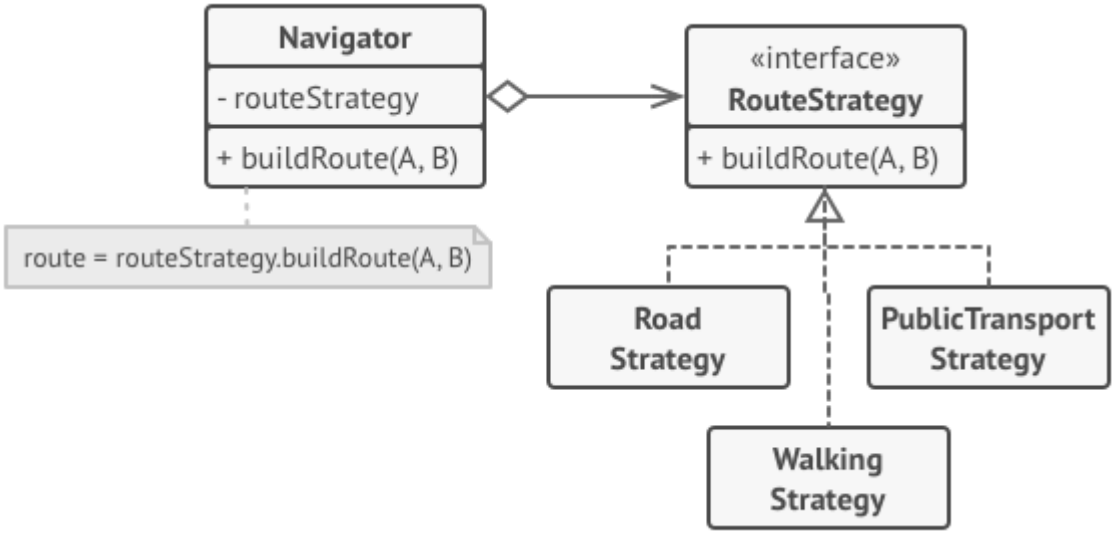

Рис 2.4 Схема взаємодії компонентів шаблона “Стратегія”

Шаблон «Пул об'єктів» (Object pool) - це породжувальний шаблон проектування, який дозволяє повторно використовувати об'єкти замість того, щоб постійно їх створювати та видаляти. Він має дві операції - взяти об'єкт із пула та повернути його назад. Якщо запит на взяття об'єкта з пула немає об'єктів потрібного типу, вони створюються. Операція повернення в пул використовується замість видалення об'єктів. Об'єкти скидають свій стан, вимикаються та повертаються назад у пул.[22]

Шаблон "Пул об'єктів" широко використовується в іграх, написаних на Unity, з таких причин:

– Кожна операція створення за допомогою Instantiate споживає багато ресурсів;
– Збірник сміття в Unity має лише одне покоління, і кожна операція збірки сміття призводить до зниження продуктивності.

Переваги:

– Значна економія обчислювальних ресурсів для створення об’єкту.
– Зменшує кількість викликів збирача сміття.

Недоліки:

– Без ручного очищення об'єкти будуть і надалі зберігатися у вимкненому стані, навіть якщо вони вже не потрібні.
– При поверненні в пул потрібно скидати всі дані про попередній стан об'єкта, щоб зробити його придатним для повторного використання.
– Якщо об'єкт містить конфіденційну інформацію, то при поверненні в пул потрібно її очистити, інакше це може призвести до ризику витоку даних.

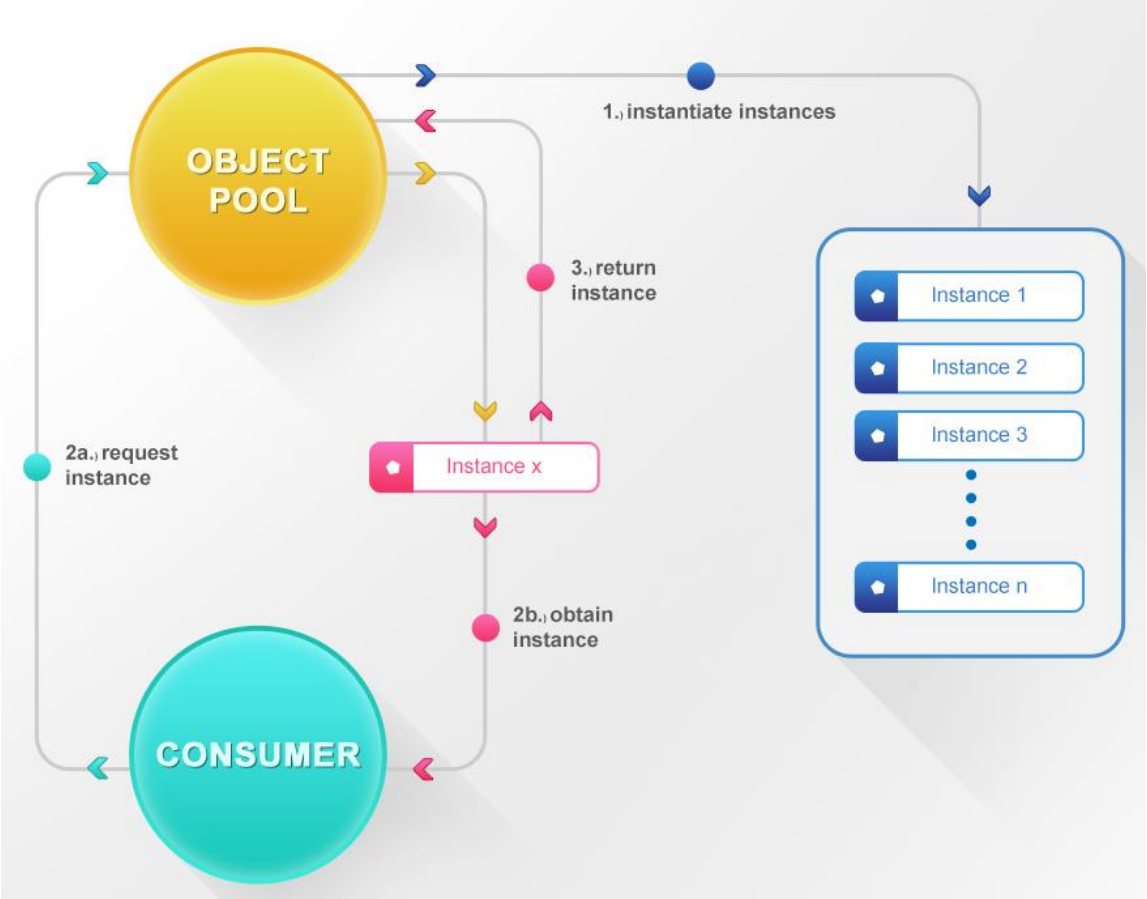

Рис 2.5 Схема взаємодії компонентів шаблона “Пул об'єктів”

Загальна схема додатку наведена на рисунку 2.6. Різні підсистеми було розподілено по кольорах, деталі їх зв’язків описано в коментарях. Представленням описаних вище шаблонів проектування є наступні елементи

ігрового додатку: EventsObserver (Спостерігач), ObjectsPooler (Пул об'єктів), ARObjectHandler (стратегія).

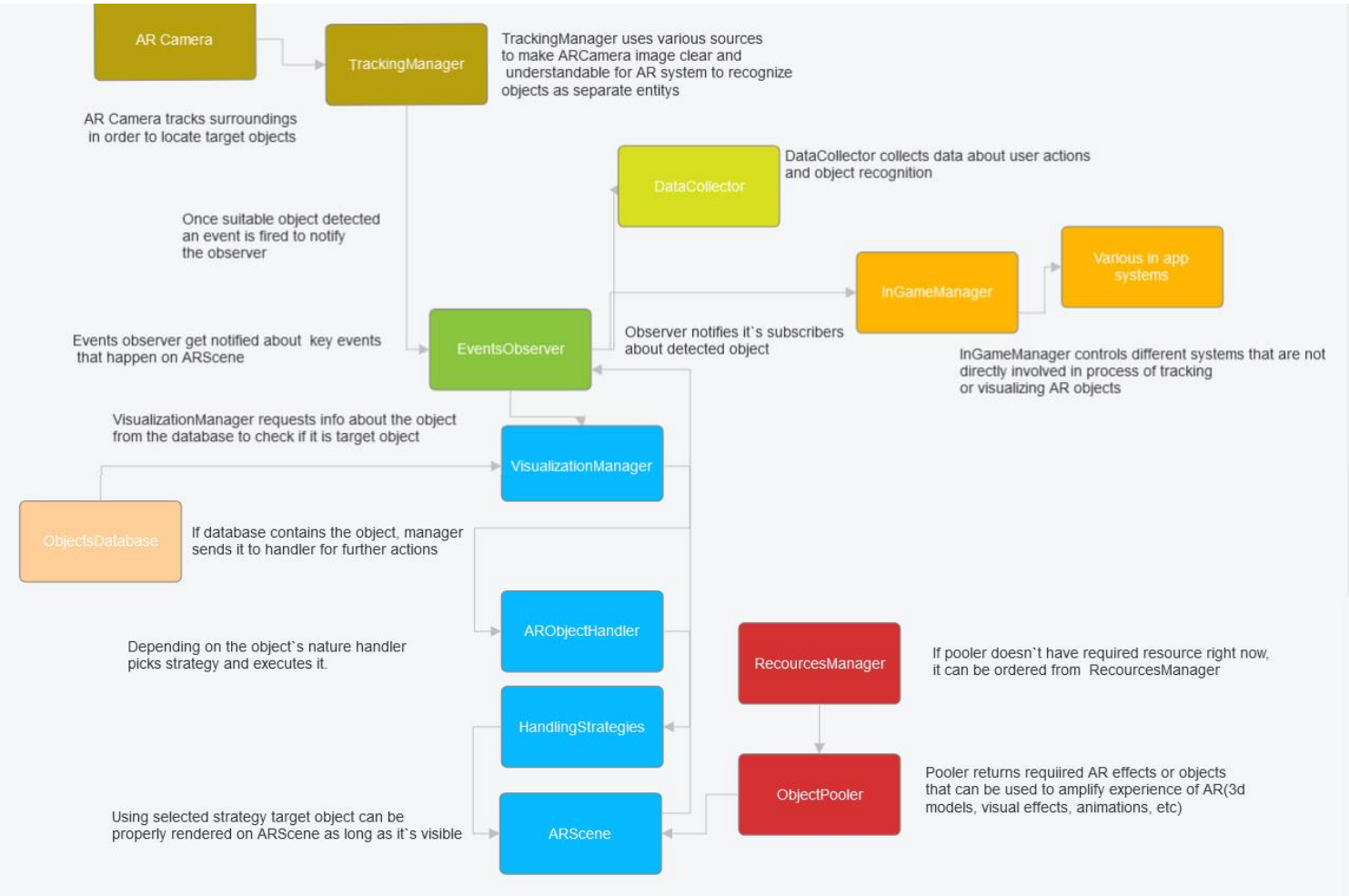

Рис 2.6 Загальна схема додатку

2.3. Опис використаних технологій та мов програмування

Unity - це відоме інтегроване середовище для розробки відеоігор та інших інтерактивних додатків. Ця платформа була створена Unity Technologies і стала однією з найпопулярніших у галузі геймдеву (розробки відеоігор). Unity використовується як для створення ігор, так і для розробки інших інтерактивних застосунків, таких як віртуальні тренажери, мультимедійні додатки, симулятори та інше. Його великий функціонал та простота використання роблять його

популярним вибором серед розробників усього світу. [22] Ось деякі ключові характеристики Unity:

- Unity дозволяє розробникам створювати додатки для різних платформ, таких як Windows, macOS, iOS, Android, Xbox, PlayStation і багато інших.
- Unity має надзвичайно потужний візуальний редактор, який дозволяє створювати ігрові об'єкти, ландшафти, анімації та інші компоненти за допомогою перетягування та відпускання. Це полегшує розробку для новачків і досвідчених розробників.
- Unity має велику бібліотеку асетів, таких як готові 3D-моделі, текстури, звуки і багато іншого, що робить процес розробки швидше та легше.
- Unity має велику та активну спільноту розробників, які допомагають один одному вирішувати проблеми, та надає безліч безкоштовних ресурсів для вивчення.
- Unity можна розширювати за допомогою плагінів, які надають додаткові можливості та функції.
- Unity підтримує розробку додатків для розширеної та віртуальної реальності, що робить його популярним в галузі AR/VR розробки.

Visual Studio 2019 - це інтегроване середовище розробки (IDE) від Microsoft для програмування на різних мовах, таких як C#, C++, Visual Basic, F#, Python, і багато інших. Це потужний інструмент, призначений для розробки різноманітних застосунків, включаючи веб-додатки, настільні програми, мобільні додатки та ігри. Visual Studio 2019 є популярним інструментом серед розробників завдяки своїм потужним можливостям, зручному інтерфейсу та підтримці великої кількості мов програмування і платформ.

Основні характеристики Visual Studio 2019 включають:

- Visual Studio 2019 підтримує ряд мов програмування, зокрема C#, C++, Visual Basic, F#, Python, JavaScript, і багато інших. Це дозволяє розробникам вибирати мову, яка найкраще відповідає їхнім потребам.
- Visual Studio має потужні інструменти відладки, включаючи можливість відстеження змін змінних, перевірку крок за кроком, відстеження викликів функцій, інтеграцію з системами контролю версій та інші.
- Visual Studio 2019 має вбудовані інструменти для розробки веб-додатків, такі як редактор HTML, CSS і JavaScript, підтримка ASP.NET, і можливості підключення до баз даних.
- Visual Studio підтримує розробку мобільних додатків для платформ Android та iOS за допомогою інструментів Xamarin.
- Visual Studio інтегрується з різними хмарними службами Microsoft, такими як Azure, що дозволяє розробникам легко підключати свої застосунки до хмарних послуг.
- Visual Studio 2019 підтримує розробку ігор за допомогою різних інструментів, включаючи Unity та Unreal Engine.
- Visual Studio включає в себе різні інструменти для аналізу якості коду, виявлення помилок та оптимізації продуктивності програм.

GitHub - це веб-платформа для зберігання та спільної роботи над проектами програмного забезпечення за допомогою системи керування версіями Git. Git дозволяє відслідковувати зміни в коді, зберігати історію версій, а також легко об'єднувати різні версії коду в єдиний проект. GitHub надає зручний інтерфейс для використання Git та додає ряд додаткових можливостей для спільної роботи над проектами. GitHub є популярним інструментом для розробників, команд програмістів та великих відкритих проектів, оскільки він дозволяє легко співпрацювати над програмним забезпеченням, відслідковувати зміни та зберігати відкритий код. Він також служить базою для великої спільноти розробників, яка обговорює нові технології та розвиває відкриті проекти.

Основні можливості та функції GitHub включають:

- Проекти на GitHub зберігаються у репозиторіях.
  Репозиторій - це сховище для вашого проекту, де зберігаються файли, теки та історія версій.
- Розробники можуть створювати відгалуження (branches) у репозиторіях для відокремленої роботи над функціями або розвитку проекту, а потім об'єднувати їх назад у головний код (main branch).
- Зміни з відгалужень можна об'єднати (merge) з головним кодом для інтеграції нових функцій чи виправлень.
- GitHub дозволяє користувачам створювати Issues для відстеження проблем, а також Pull Requests для пропонування змін в основний код. Це спрощує обговорення та обговорення змін в коді перед їх включенням у проект.
- Вбудовані автоматизовані засоби для тестування, збирання інформації та розгортання коду прямо на платформі GitHub.
- Можливість розміщення статичних веб-сайтів безкоштовно за допомогою GitHub.
- Розробники можуть спільно працювати над проектами та керувати рівнями доступу, дозволяючи або обмежуючи права інших користувачів до репозиторіїв.

Vuforia - це платформа розпізнавання образів та розширеної реальності (AR), яка дозволяє розробникам створювати інтерактивні додатки для мобільних пристроїв та інших AR-сумісних пристроїв. Розроблена компанією PTC, Vuforia забезпечує можливість розпізнавання об'єктів в реальному часі та належну їх інтеграцію з віртуальним контентом, що відображається на екрані пристрою користувача. Vuforia знаходить широке застосування у відкритих проектах AR, іграх, додатках для маркетингу та навчанні, де взаємодія з фізичними об'єктами в реальному часі є ключовою функцією.

Основні характеристики та можливості Vuforia:

- Vuforia може розпізнавати образи, маркери, QR-коди та 3D об'єкти у реальному часі. Це дозволяє додавати віртуальний контент до фізичних об'єктів або поверхонь.
- Розробники можуть створювати спеціальні маркери або визначати області, які можуть бути використані для розпізнавання та взаємодії з AR-контентом.
- Vuforia надає плагіни та інструменти для інтеграції з популярним двигуном гри Unity. Це дозволяє розробникам створювати AR-додатки для різних платформ, включаючи iOS, Android, та HoloLens.
- Платформа може відстежувати рухи користувача та розпізнавати жести, що дозволяє взаємодіяти з AR-сценами за допомогою жестів та рухів.
- Vuforia підтримує розширену реальність на великих відстанях(Extended Range AR), що дозволяє взаємодіяти з великими об'єктами та сценами на значній відстані від пристрою.
- Vuforia може розпізнавати та аналізувати рельєфні особливості поверхні(Smart Terrain), що дозволяє створювати інтерактивні сцени в реальному часі.
- Vuforia може створювати області визначення(Area Targets), що дозволяє розпізнавати та взаємодіяти з великими просторовими зонами, такими як кімнати чи ландшафти.

Adobe Photoshop - це відомий графічний редактор, розроблений компанією Adobe Inc. Ця програма дозволяє редагувати та маніпулювати растровими зображеннями з вражаючою гнучкістю та функціональністю. Photoshop є стандартом у галузі графічного дизайну, фотографії та візуальних мистецтв. Adobe Photoshop доступний як частина платіжної підписки Adobe Creative Cloud та включає в себе безліч корисних інструментів для креативної роботи з зображеннями та графікою.

Основні функції та можливості Adobe Photoshop включають:

- Photoshop дозволяє коригувати кольори, яскравість, контраст, насиченість, розмиття, гаму та інші параметри зображення. Різноманітні інструменти дозволяють видаляти небажані елементи, виправляти дефекти та застосовувати різні ефекти.
- Photoshop дозволяє створювати графічні елементи для веб-сайтів, ілюстрації, банери, логотипи, плакати та інші графічні проекти. Він має інструменти для малювання, редагування зображень, текстурингу та інші можливості для створення графічних мистецтв.
- Photoshop є популярним інструментом для обробки фотографій. Він має функції ретушування, реконструкції, зміни розміру, зменшення шуму, роботи з RAW-файлами та інші засоби для поліпшення якості фотографій.
- Photoshop використовує концепцію шарів, що дозволяє редагувати різні елементи зображення незалежно один від одного. Маски шарів дозволяють точно визначити, які частини зображення відображаються, а які - ні.
- Photoshop дозволяє вставляти та редагувати текст на зображенні. Він має різноманітні інструменти для вибору шрифтів, кольорів, розмірів та інших параметрів тексту.
- Photoshop легко інтегрується з іншими продуктами Adobe, такими як Adobe Illustrator та Adobe InDesign, що дозволяє спростити роботу з графічними проєктами.

Xcode - це інтегроване середовище розробки (IDE) для програм на платформах Apple, таких як iOS, macOS, watchOS та tvOS. Це офіційний інструмент для розробки програмного забезпечення для пристроїв, вироблених компанією Apple.

Ось деякі ключові риси та можливості Xcode:

- Xcode підтримує ряд мов програмування, таких як Swift, Objective-C та AppleScript. Swift є сучасною мовою програмування, спеціально розробленою для розробки програм для платформ Apple.

- За допомогою Xcode ви можете створювати додатки для iPhone, iPad, Mac та інших пристроїв Apple. Інтерфейс Xcode дозволяє вам відстежувати різні пристрої та емулятори для тестування додатків.
- Interface Builder є інструментом для візуального створення інтерфейсу користувача для додатків. Його інтеграція з Xcode дозволяє розробникам легко створювати та налаштовувати елементи інтерфейсу.
- Xcode надає розширені можливості для дебаггінгу коду та профілювання додатків. Розробники можуть використовувати інструменти, такі як Instruments для аналізу продуктивності та виявлення проблем.
- Xcode інтегрований з системами управління версіями, такими як Git, що спрощує роботу в команді та ведення версій свого коду.
- Розробники можуть розширювати функціональність Xcode за допомогою різних плагінів та розширень.
- З Xcode ви можете розробляти додатки, які працюють на різних пристроях та платформах Apple, використовуючи спільний код.
- Xcode має вбудовані інструменти для автоматизованого тестування додатків, що дозволяє розробникам створювати стабільні та надійні програми.

C# (вимовляється як "Сі-шарп") - це об'єктно-орієнтована мова програмування, розроблена компанією Microsoft. Вона є однією з основних мов для розробки програм для платформи Microsoft .NET. C# є однією з популярних мов програмування у світі, особливо серед розробників, які працюють у середовищі Microsoft. Її зручний синтаксис, об'єктно-орієнтовані можливості та безпека роблять її важливим інструментом для створення різноманітних застосунків.[23]

Ось деякі ключові аспекти та характеристики мови програмування C#:

- C# спадковий від мови C і C++, але має спрощений та більш безпечний синтаксис. Вона надає велику кількість вбудованих типів даних та конструкцій мови для зручного програмування.

- C# - це повністю об'єктно-орієнтована мова програмування. Всі елементи в C# є об'єктами, що сприяє відновленню об'єктно-орієнтованих концепцій, таких як спадкування, інкапсуляція і поліморфізм.
- C# автоматично керує пам'яттю, використовуючи систему сміття (garbage collection), що зменшує ризик витоку пам'яті та полегшує програмування.
- C# підтримує багато функцій з бібліотек, що спрощують розробку програм, таких як робота з базами даних, робота з графікою, мережеві операції та інше.
- C# має вбудовані засоби для перевірки безпеки, такі як управління типами, що роблять код менш вразливим до атак типу "buffer overflow".
- Хоча C# розроблена для платформи Windows, існують інструменти, такі як Mono та .NET Core, які дозволяють використовувати C# на інших платформах, включаючи Linux та macOS.
- C# інтегрується з іншими технологіями Microsoft, такими як Windows Forms, WPF (Windows Presentation Foundation) та ASP.NET для розробки десктопних та веб-застосунків.
- Мова постійно розвивається, з'являються нові функції та підтримка для сучасних технологій, таких як обробка паралельних процесів, асинхронні операції, LINQ (Language Integrated Query) та багато іншого.

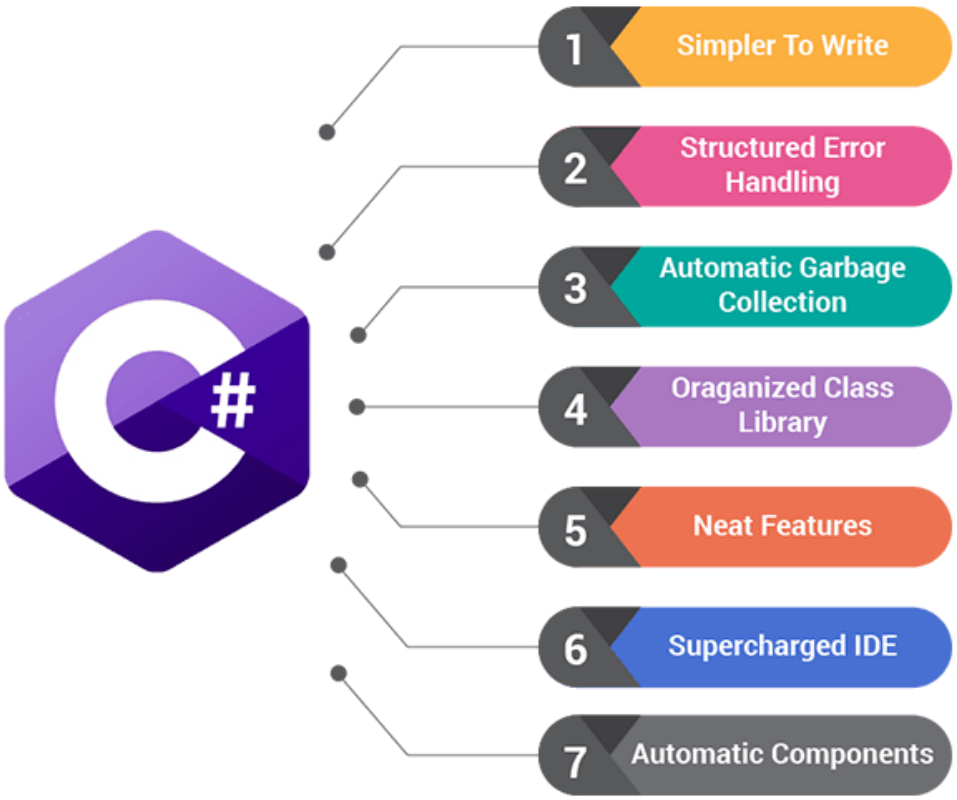

Рис 2.7 Переваги мови програмування C#

Windows 11 - це операційна система, розроблена корпорацією Microsoft, яка є наступником Windows 10. Перед виходом Windows 11 Microsoft заявила, що це оновлення принесе новий дизайн, поліпшену продуктивність та ряд нових функцій. Ось деякі ключові аспекти Windows 11:

- Windows 11 представляє оновлений дизайн з покращеними елементами керування та зміненим меню "Пуск". Дизайн отримав назву "Sun Valley" і включає в себе округлі кути вікон, новий шрифт, покращені іконки та інші естетичні зміни.
- Меню "Пуск" стало центральним елементом завдяки централізованому розташуванню піктограм та покращеному пошуку. Таскбар тепер

розташований по центру за замовчуванням, але може бути переміщений назад до лівого краю для тих, хто звик до такого розташування.

- Windows 11 включає новий віджетний панель, яка надає інформацію про новини, погоду, календар та інші корисні дані.
- Магазин Windows отримав перегляд та поліпшення, що включає новий інтерфейс користувача та покращену навігацію.
- Windows 11 отримав підтримку для Android-додатків, що дозволить користувачам запускати їх на своїх комп'ютерах.
- Windows 11 отримав підтримку DirectX 12 Ultimate, що може поліпшити графічні можливості для геймерів та творців контенту.
- Windows 11 має покращене відтворення HDR-контенту, Dolby Atmos та Dolby Vision.
- Змінено роботу віртуальних столів та покращено мультитаскінг.

Вимоги до апаратного забезпечення:

- Процесор з тактовою частотою мінімум 1 ГГц, який має щонайменше два ядра.
- Мінімум 4 ГБ оперативної пам’яті.
- Мінімум 64 ГБ місця у сховищі.
- Системна прошивка UEFI з підтримкою безпечного завантаження.
- Відеоадаптер, сумісний з DirectX 12 або новішою версією з драйвером WDDM 2.0.

Android Studio - це інтегроване середовище розробки (IDE), створене спеціально для розробників Android-додатків. Розроблено компанією Google, Android Studio надає потужні інструменти та ресурси для створення якісних мобільних додатків для платформи Android. Android Studio - це стандартне середовище розробки для більшості Android-розробників і надає потужні інструменти для розробки високоякісних та інноваційних мобільних додатків для платформи Android.

Ось деякі ключові аспекти та функції Android Studio:

- Android Studio має зручний та користувацьки-орієнтований інтерфейс користувача. Вона має інтелектуальний кодовий редактор з функцією автодоповнення та можливостями рефакторингу коду. Редактор також надає можливості перегляду різних файлів, включаючи макети, ресурси, маніфести та інші.
- Android Studio має вбудовані емулятори Android, які дозволяють розробникам перевіряти свої додатки на різних версіях Android та на різних типах пристроїв. Також є можливість підключення реальних Android-пристроїв для тестування додатків.
- Android Studio сприяє легкості реалізації та локалізації додатків. Розробники можуть легко додавати нові екрани, макети та функціональність до своїх додатків.
- Android Studio оновлюється з кожним випуском нової версії Android, що дозволяє розробникам використовувати найновіші можливості та API, які надає платформа Android.
- Можливість створення графічних інтерфейсів за допомогою drag-and-drop інструментів. Також можна легко керувати ресурсами, включаючи зображення, макети, кольори, рядки та інші.
- Android Studio інтегрується з іншими корисними інструментами, такими як системи керування версіями (наприклад, Git), що полегшує співпрацю та розробку у команді.
- Android Studio оптимізована для швидкої роботи та забезпечення високої продуктивності під час розробки додатків.

Blender - це потужний та безкоштовний(з відкритим вихідним кодом) графічний пакет для моделювання 3D-графіки, анімації, рендерингу, композитингу, редагування відео та інших задач, пов'язаних з візуалізацією. Основна особливість Blender полягає в тому, що він є повністю безкоштовним та розповсюджується під вільною ліцензією GNU General Public License.

Основні риси та можливості Blender:

- Blender надає широкі можливості для створення 3D-моделей. Він підтримує різні методи моделювання, такі як полігональне, субдивізійне, та метаболічне моделювання.
- Blender має потужні інструменти для створення анімацій. Ви можете створювати складні анімації для об'єктів, персонажів, частинок, та багато іншого.
- Вбудовані двигуни рендерингу, такі як Cycles та Eevee, дозволяють створювати вражаючі візуальні ефекти та фотореалістичні зображення.
- Blender надає інструменти для редагування та комбінування різних елементів, а також для додавання спеціальних ефектів та корекції зображень.
- Інструменти скульптування дозволяють художникам створювати деталізовані 3D-моделі.
- Blender підтримує створення різноманітних спеціальних ефектів, таких як симуляція частинок, рідин, диму, огню та інших.
- Blender має унікальну інтерфейсну систему, яка може здатися складною для новачків, але є потужною та ефективною для досвідчених користувачів.
- Заснований на відкритому вихідному коді, Blender підтримує розширюваність через плагіни та додатки, що дозволяє розширювати його функціонал.

Substance Painter - це програмне забезпечення для текстурування та рендерингу 3D-моделей, яке виробляється компанією Allegorithmic, яка зараз є частиною Adobe. Це потужний інструмент для художників, які працюють у сфері візуального мистецтва, геймдизайну, відео та анімації. Ось деякі ключові характеристики Substance Painter:

- Substance Painter працює відповідно до принципів фізично заснованого рендерингу, що дозволяє створювати реалістичні матеріали з точними фізичними властивостями.

- Він пропонує зручний інтерфейс для роботи з текстурами в реальному часі. Ви можете наносити текстури з різних матеріалів і шарів, користуючись різними пензлями та інструментами.
- Substance Painter використовує концепцію Smart-шарів, що дозволяє зберігати параметри матеріалів у шарах для подальшого редагування та легкої зміни текстур.
- Ви можете створювати текстурні карти для різних цілей, таких як колір, нормалі, висота, грубість і інші. Substance Painter автоматично генерує карти для PBR-рендерингу.
- Substance Painter може взаємодіяти з іншими програмами для 3D-моделювання, такими як Autodesk Maya, Blender, Unity, Unreal Engine тощо. Ви можете вживати зміни, зроблені в Substance Painter, в реальному часі.
- Програма дозволяє створювати різні типи шарів, такі як кольорові, глибинні, висотні, рельєфні тощо, для досягнення різноманітності та деталізації.
- Підтримка візуального ефекту переосвітлення (Real-Time IBL) дозволяє переглядати зміни в освітленні в реальному часі.

RizomUV - це програмне забезпечення для розгортання UV-карт, яке використовується у візуальному мистецтві, геймдизайні, анімації та інших галузях, пов'язаних із створенням 3D-моделей. Основний фокус RizomUV полягає в ефективному та швидкому процесі створення і редагування UV-розгортань, що є важливим етапом при створенні текстур для 3D-моделей. Ось деякі ключові характеристики RizomUV:

- RizomUV надає інструменти для швидкого та ефективного розгортання UV-карт для складних 3D-моделей. Користувачі можуть взаємодіяти з об'єктами та їхніми UV-розгортаннями в реальному часі.

- RizomUV може працювати з різними форматами файлів для обміну даними між іншими програмами для 3D-моделювання.
- Програма намагається максимізувати використання текстурного простору, дозволяючи оптимізувати розміщення елементів на UV-карті для оптимального використання текстур.
- RizomUV дозволяє редагувати та виправляти UV-розгортання в реальному часі, щоб виправити будь-які проблеми, які можуть виникнути в процесі створення 3D-моделі.
- Програма може ефективно обробляти великі моделі та великі об'єми даних, що робить її придатною для професійного використання.
- RizomUV може працювати з UDIM-розгортанням, яке дозволяє створювати текстури для об'єктів, що охоплюють декілька картах (тайлах), а також із 4D-контекстами, що дозволяє працювати з часовими аспектами UV-розгортання.
- RizomUV може взаємодіяти з різними програмами для 3D-моделювання та текстурування, щоб спростити обмін даними та використання UV-розгортань в інших контекстах.

Marmoset Toolbag 4 - це програмне забезпечення для візуалізації та рендерингу 3D-сцен, яке призначене для використання в індустрії відтворення ігор, графічного дизайну, архітектури та інших областей, де важливий високоякісний візуальний контент. Ось деякі ключові особливості Marmoset Toolbag 4:

- Marmoset Toolbag пропонує рендеринг у реальному часі, що дозволяє користувачам миттєво бачити зміни у своїх проектах без необхідності очікування на завершення великих рендерингових завдань.
- Програма підтримує відтворення різних типів матеріалів, включаючи металеві, дерево, шкіру, скло та інші. Це дозволяє створювати реалістичні візуалізації з різноманітними поверхнями.

- Інтерфейс Marmoset Toolbag призначений для зручності користувача та включає інструменти для легкого доступу до основних функцій, таких як освітлення, матеріали та текстури.
- Програма підтримує різні техніки освітлення, такі як Global Illumination (GI), Image-Based Lighting (IBL) та інші. Це дозволяє створювати динамічне та реалістичне освітлення сцен.
- Marmoset Toolbag дозволяє створювати анімації та взаємодію з об'єктами у реальному часі. Це допомагає в створенні динамічних та привабливих сцен.
- Програма забезпечує високоякісний рендеринг, що дозволяє отримати зображення високої роздільності з деталізованими текстурами та ефектами.
- Marmoset Toolbag підтримує виведення зображень та відео у різні формати, що спрощує обмін результатами та інтеграцію з іншими програмами.
- Програма підтримує віртуальну реальність (VR), що робить її використовуваною для проектів, пов'язаних із віртуальною реальністю та ігровою розробкою.

2.4. Висновки з другого розділу

В даному розділі були проаналізовані методи, що стоять за роботою систем доповненої реальності. Особливу увагу слід приділити методам комп’ютерного зору та глибинного навчання. Саме вони в комбінації з іншими методами дозволяють забезпечити реальність доданого простору на рівні, що нещодавно був просто неможливий. Також були розглянуті архітектурний підхід та шаблони проєктування, що були використані під час розробки даного проєкту. Використанні для розробки додатку технології були описані детально, з урахуванням ключових особливостей.

РОЗДІЛ 3.
РОЗРОБКА ТА АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Обґрунтування використаних технологій

Перед початком розброки ігрового додатку слід було обрати ігровий двигун та платформу для доданої реальності. В якості двигуна мною були розглянуті наступні опції: UnrealEngine, Unity, Godot. Для AR платформи були розглянуті такі варіанти: Pikkard, Vuforia, Kudan та Maxst. Зрештою я зупинився на тандемі Unity та Vuforia з причин, що будуть описані нижче.

Unity та Vuforia — це мікс двох потужних інструментів, які разом утворюють ідеальний альянс для розробки захоплюючих ігор, сповнених доповненої реальності (AR). Unity, визнана своєю величезною спільнотою розробників та гнучкістю, володіє всіма необхідними інструментами для створення вражаючих ігрових вирішень. Її мультиплатформеність дозволяє легко переймати ігровий витвір з iOS на Android і навпаки, що розширює потенційну аудиторію додатку.

AR Foundation вбудована в Unity, спрощує розробку для різних AR-платформ, забезпечуючи єдиний код для ARKit та ARCore. Швидкість розробки підтримується інтуїтивно зрозумілим інтерфейсом, дозволяючи розробникам швидко прототипувати та впроваджувати ідеї.

У свою чергу, Vuforia, як платформа для розширеної реальності, вносить свій унікальний внесок у розпізнавання об'єктів. Здатність розпізнавати зображення, текст, маркери та 3D об'єкти в реальному часі відкриває безмежні можливості для створення захоплюючих AR-сценаріїв. Ця надійність розпізнавання робить Vuforia ідеальним партнером для Unity, доповнюючи його можливості.

Такий комбінований підхід дозволяє розробникам не тільки створювати ігри з вражаючою графікою та інтерактивністю, але також використовувати потужності доповненої реальності для того, щоб занурити гравців у

захоплюючий віртуальний світ. Гнучкість, швидкість розробки та висока якість розпізнавання об'єктів роблять Unity та Vuforia неперевершеним дуетом для розробників, які бажають втілити свої ідеї у світ AR-ігор.

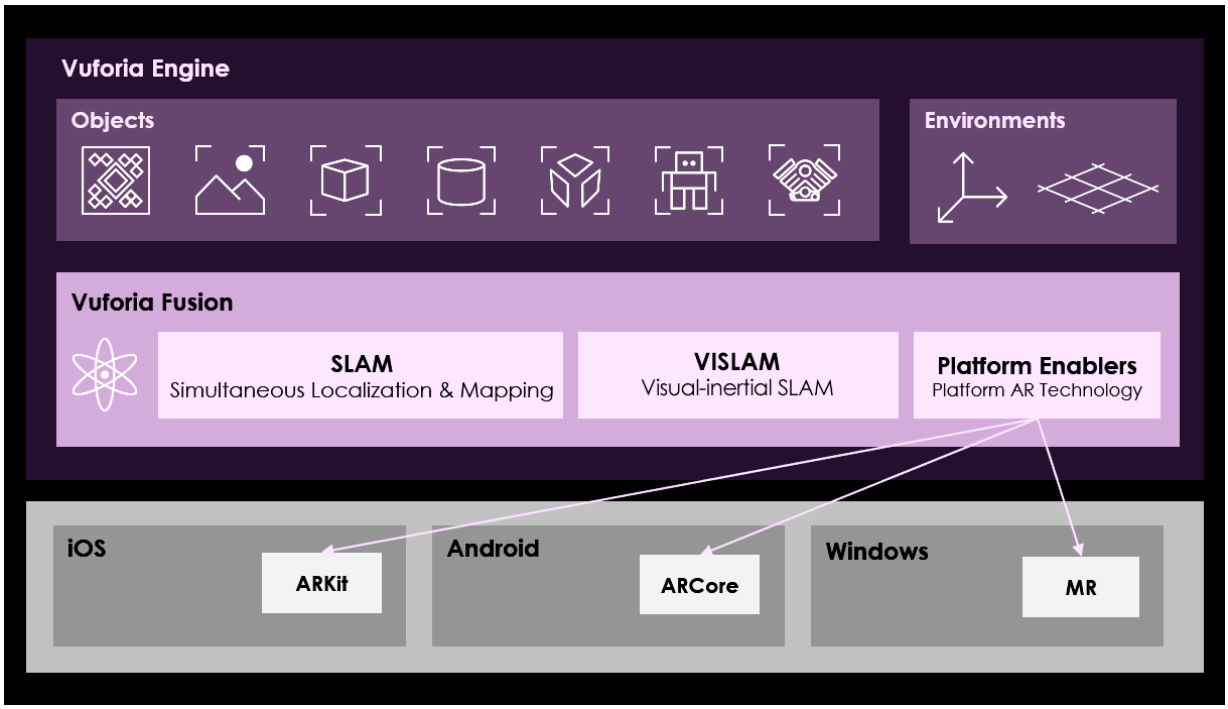

Рис 3.1 Схематичне зображення можливостей Vuforia

3.2 Технічні вимоги для запуску додатку

Перш ніж встановити додаток на ваш пристрій варто перевірити чи відповідає він мінімальним системним вимогам. Для різних платформ вони є наступними:

Android:

Операційна система: Android 8.0 або більше нова.

Процесор: Quad-core або новіший.

Оперативна пам'ять: Мінімум 2 ГБ RAM.

Камера: Задня камера з підтримкою фокусування та забезпеченням високої якості зображення.

Вільне місце: 100мб+

iOS:

Операційна система: iOS 15 або більше нова.

Процесор: A9 x64 або новіший.

Оперативна пам'ять: Мінімум 2 ГБ RAM.

Камера: Задня камера з підтримкою фокусування та забезпеченням високої якості зображення.

Вільне місце: 100мб+

3.3 Встановлення ігрового додатку на пристрій

Android:

Необхідно завантажити та запустити .apk файл. Далі з’явиться вікно встановлення додатків, дивіться рис. 3.2. (перед цим може знадобитися дозволити встановлення програм з невідомих джерел). Після встановлення .apk файлу додаток запускається за допомогою іконки на головному екрані або зі списку додатків.

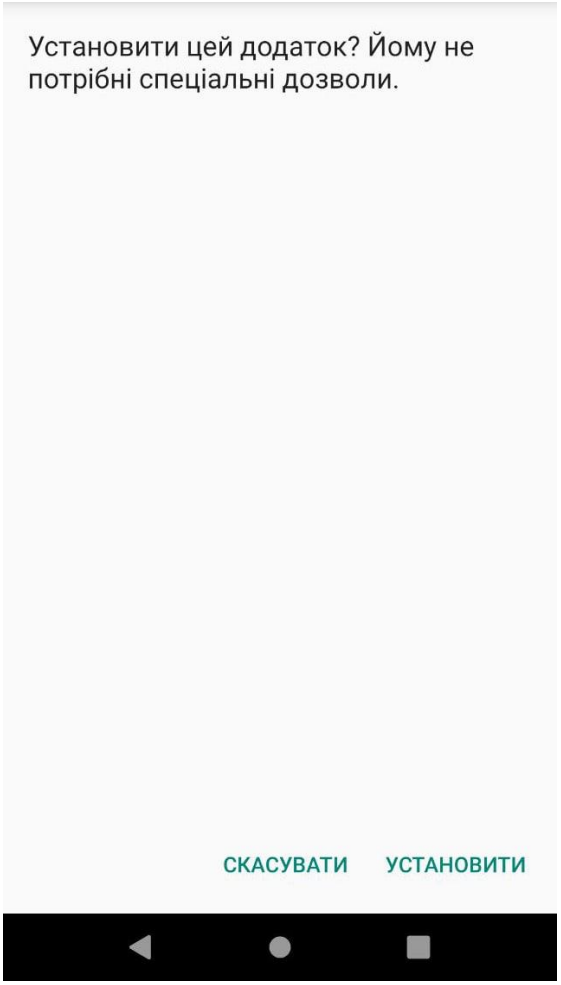

Рис 3.2 Екран встановлення додатку

IOS:

У випадку пристроїв компанії Apple єдиний шляхом, крім отримання з магазину додатків, є використання програми TestFlight для встановлення тестових додатків. Для цього слід завантажити програму з магазину і отримати запрошення на тестування додатку. В результаті ви побачите вікно як на Рис. 3.3. На цьому ж екрані можна встановити попередні версії програми або надіслати розробникам свій зворотній зв’язок.

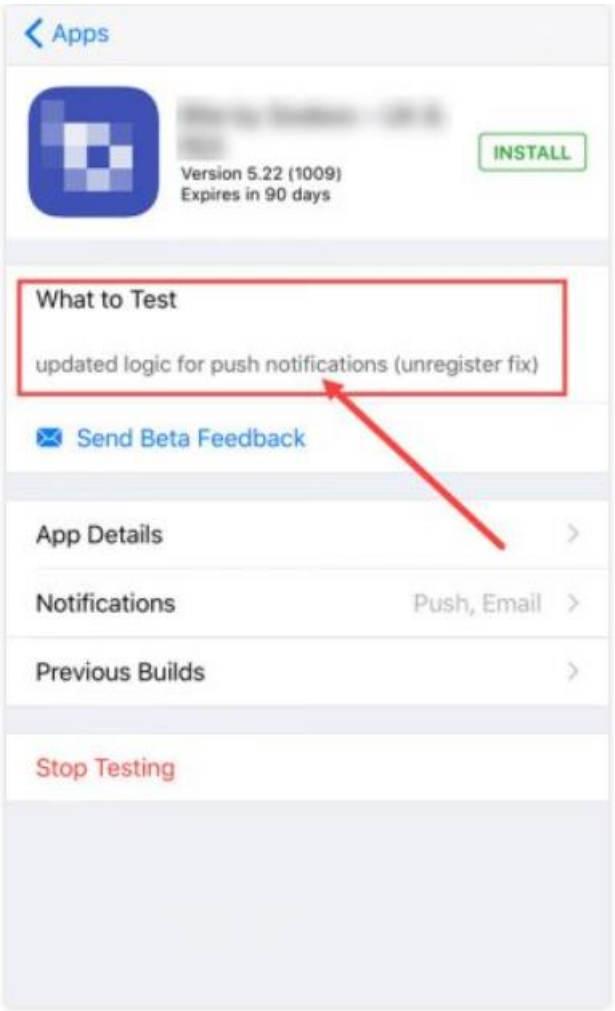

Рис. 3.3 Екран встановлення додатку в TestFlight

3.4 Опис інтерфейсу користувача

Після встановлення і відкриття ігрового додатку першим екраном, що бачить користувач є головне меню. Меню надає можливості запустити

гру, перейти в режим скану, відкрити екран налаштувань, переглянути колекцію відкритих карток або вийти з гри, дивіться рис. 3.4 - рис. 3.8.

Після завершення гри користувачу буде показано екран кінця гри з результатами партії, дивіться рис. 3.9

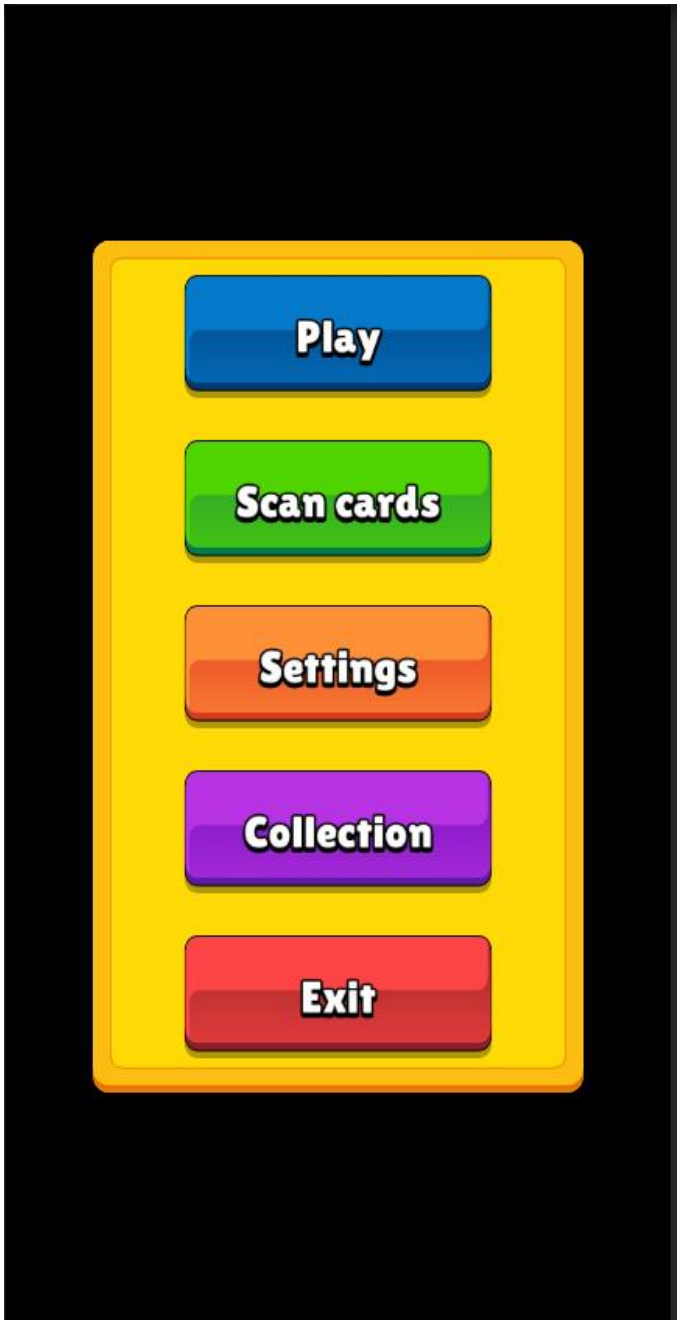

Рис. 3.4 Екран головного меню

Рис. 3.5 Ігровий екран

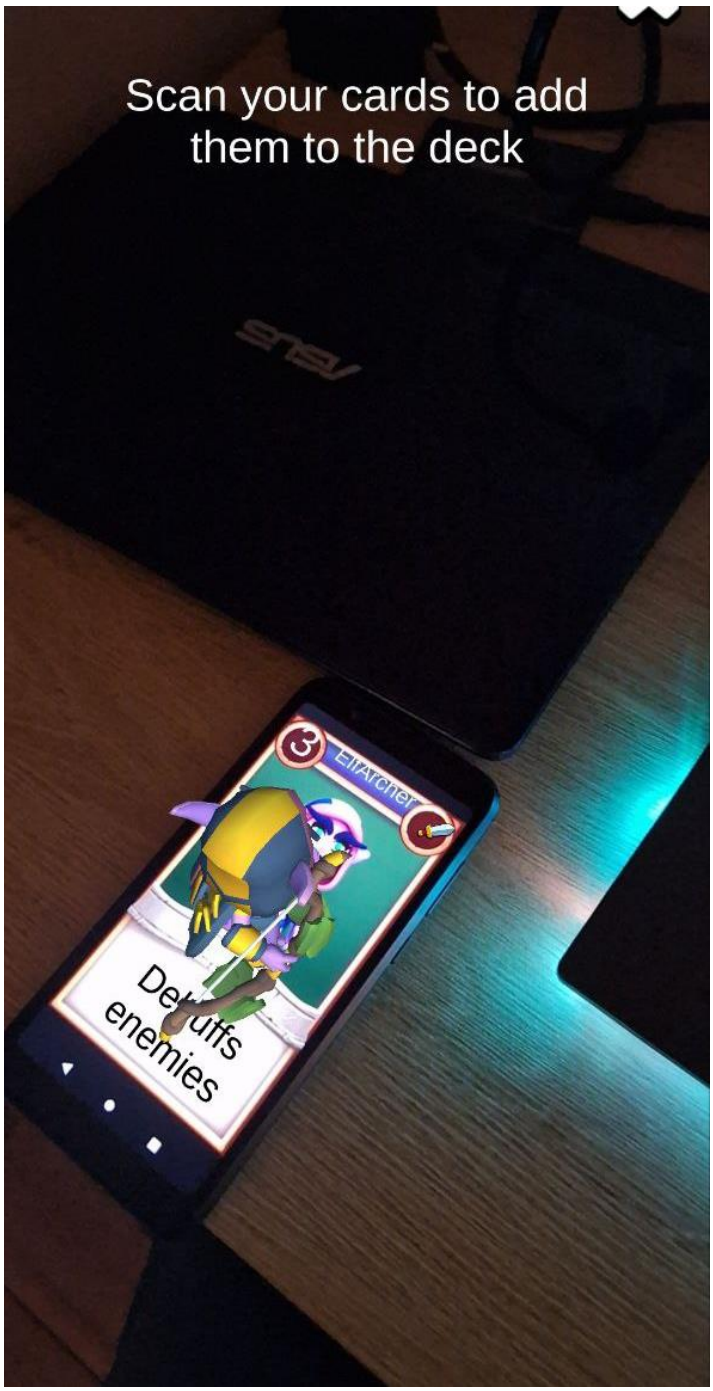

Рис. 3.6 Екран сканування

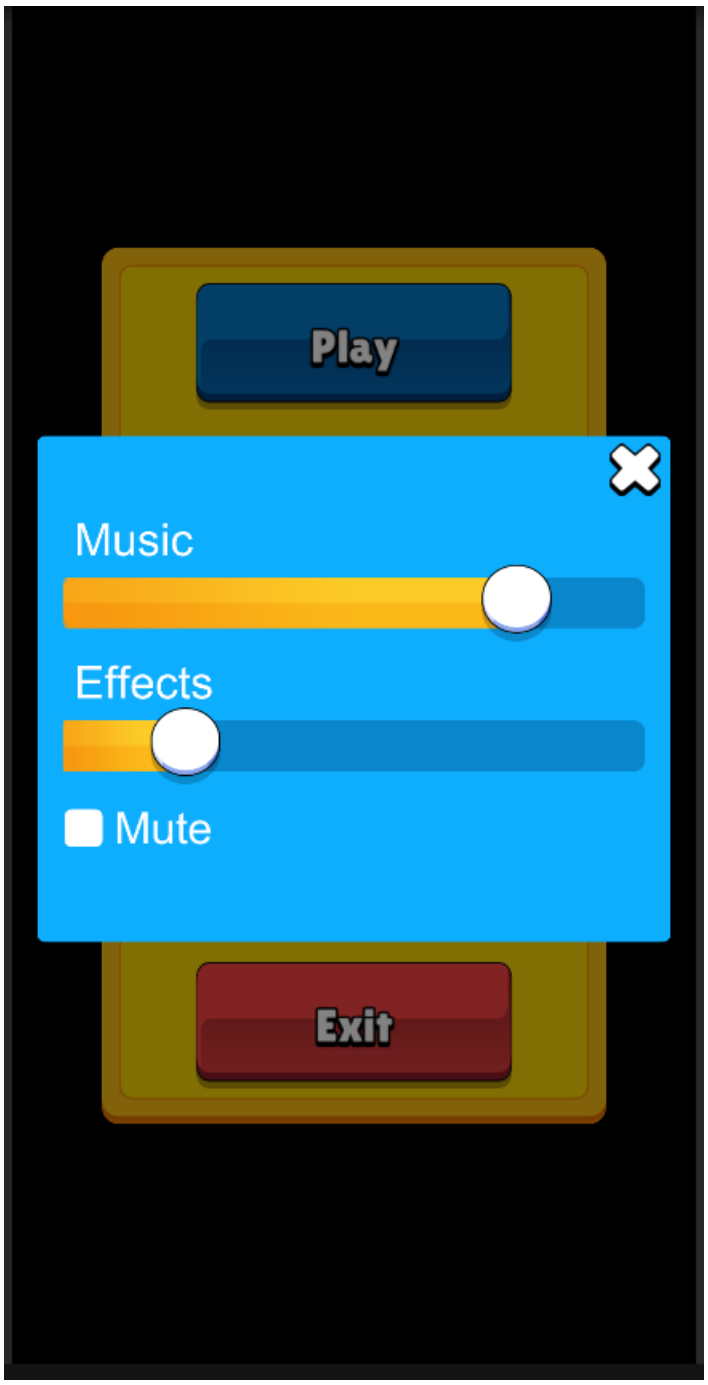

Рис. 3.7 Екран налаштувань

Рис. 3.8 Екран колекції

Рис. 3.9 Екран кінця гри

3.5 Опис та аналіз вихідних даних

В процесі використання ігрового додатку збираються дані для подальшого аналізу за наступними напрямами:

– скільки разів було запущено той чи інший екран;
– скільки часу користувач провів на кожному екрані;
– швидкість розпізнавання об’єкту та відмальовка доданої реальності навколо нього;
– яка картка і скільки разів була використана в грі;
– результат завершення партії.

Зібрані дані серіалізуються в текстовий документ формату JSON (JavaScript Object Notation), що зберігається на пристрої і потім може бути зручно переведений в таблицю Microsoft Excel для подальшого аналізу.

На рис. 3.10 можна побачити результати тестового запуску програми.

Id	result
MainMenu	3.0
Time spent in MainMenu	21.7
Settings	3.0
Time spent in Settings	16.606
Collection	1.0
Time spent in Collection	0.830
ScanCards	1.0
Drone recognized in	0.638
Astronaut recognized in	0.001
Oxygenrecognized in	0.008
Time spent in ScanCards	0.986
Game	1.0

Рис. 3.10 Результати запуску програми

Як видно найбільше часу було проведено на екрані головного меню, оскільки воно є зв’язуючим між іншими екранами, в той же час менше за все часу було витрачено на екрані колекцій, оскільки на початку використання він ще є досить пустим.

Результати відстеження карток теж є досить цікавими. Так найбільше часу пішло на першу картку, що пов’язано з необхідністю ініціалізації системи та фокусу камери. Решта карток була розпізнана значно швидше, завдяки підготованості користувача і системи то подальшого сканування. Також на швидкість розпізнавання можуть впливати такі фактори як якість камери пристрою, що використовується, освітлення, складність об’єкту, що зараз проходить процес розпізнавання та поведінка користувача (тряска камери чи рух ускладнять фокус камери і розпізнавання об’єкту).

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблене програмне забезпечення, що являє собою мультиплатформений ігровий додаток з використанням доповненої реальності для застосування на мобільних платформах під різним ОС, зібрано та проаналізовано результати використання програми. З дослідницької сторони роботи було проведено дослідження засобів та принципів взаємодії з AR, оцінено їх переваги та недоліки та обрано найбільш придатні для використання в мобільних ігрових додатках, що має потенціал для використання іншими розробниками задля зменшення строків та витрат на розробку власних проектів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Forgotten genius: the man who made a working VR machine in 1957 URL:https://www.techradar.com/news/wearables/forgotten-genius-the-man-who-made-a-working-vr-machine-in-1957-1318253/2 (дата звернення: 6.10.2023).

2. Rosenberg, L.B. (1993). "Virtual Fixtures: Perceptual Overlays for Telerobotic Manipulation". In Proc. of the IEEE Annual Int. Symposium on Virtual Reality (1993): pp. 76–82.

3. “A Survey of Augmented Reality” URL: http://www.cs.unc.edu/~azuma/ARpresence.pdf (дата звернення: 7.10.2023).

4. Santoni, F.; De Angelis, A.; Moschitta, A.; Carbone, P. MagIK: A Hand-Tracking Magnetic Positioning System Based on a Kinematic Model of the Hand. IEEE Trans. Instrum. Meas. 2021, 70, 1–13.

5. Heidemann, G.; Bax, I.; Bekel, H. Multimodal interaction in an augmented reality scenario. In Proceedings of the 6th International Conference on Multimodal Interfaces, State College, PA, USA, 13–15 October 2004; pp. 53–60.

6. Wang, X.; Dunston, P.S. Comparative effectiveness of mixed reality-based virtual environments in collaborative design. IEEE Trans. Syst. Man Cybern. Part C 2011, 41, 284–296.

7. Danielsson, O.; Holm, M.; Syberfeldt, A. Augmented reality smart glasses for operators in production: Survey of relevant categories for supporting operators. Procedia CIRP 2020, 93, 1298–1303.

8. Gibson, L.; Hanson, V.L. Digital motherhood: How does technology help new mothers? In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Paris, France, 27 April–2 May 2013; pp. 313–322.

9. Henrysson, A.; Ollila, M. UMAR: Ubiquitous mobile augmented reality. In Proceedings of the 3rd International Conference on Mobile and

Ubiquitous Multimedia, College Park, MD, USA, 27–29 October 2004; pp. 41–45.

10. Höllerer, T.; Wither, J.; DiVerdi, S. “Anywhere augmentation”: Towards mobile augmented reality in unprepared environments. In Location Based Services and TeleCartography; Springer: Berlin/Heidelberg, Germany, 2007; pp. 393–416.

11. Evennou, F.; Marx, F. Advanced integration of WiFi and inertial navigation systems for indoor mobile positioning. EURASIP J. Adv. Signal Process. 2006, 2006, 1–11.

12. M. Agrawal, K. Konolige, and M. R. Blas. CenSurE: center surround extremas for realtime feature detection and matching. In ECCV, pages 102–115, 2008.

13. S.Benhimane and E. Malis. Homography-based 2D visual tracking and servoing. I. J. Robotic Research, 26(7):661–676, 2007.

14. M. Donoser, P. Kontschieder, and H. Bischof. Robust planar target tracking and pose estimation from a single concavity. In ISMAR, pages 9–15, 2011.

15. M. Pupilli and A. Calway. Real-time camera tracking using known 3D models and a particle filter. In ICPR, pages 199– 203, 2006

16. Occlusion Matting: Realistic Occlusion Handling for Augmented Reality URL:https://www.researchgate.net/publication/321257480_Occlusion_Matting_Realistic_Occlusion_Handling_for_Augmented_Reality_Applications(дата звернення 25.10.2023)

17. Richard Hartley and Andrew Zisserman (2003). Multiple View Geometry in computer vision. Cambridge University Press.

18. Humpherys, Jeffrey (2012). "A Fresh Look at the Kalman Filter". SIAM Review. 54 (4): 801–823

19. Convolutional Neural Networks in Visual Computing , Ragav Venkatesan, Baoxin Li. Boca Raton, 2017.128p

20. Observer pattern. URL: https://refactoring.guru/design-patterns/observer (дата звернення: 1.11.2023).

21. StrategyPattern. URL: https://refactoring.guru/design-patterns/strategy (дата звернення: 1.11.2023).

22. ObjectPooler. URL: https://sourcemaking.com/design_patterns/object_pool (дата звернення: 1.11.2023).

23. Hocking J. Unity in Action, Third Edition: Multiplatform game development in C#. Manning, 2021. 416 p.

24. Whitaker R. The C# Player's Guide (5th Edition). Starbound Software, 2022. 495 p.

ДОДАТОК А

ЛІСТИНГ ПРОГРАМИ

AIPlayer.cs

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

public class AIPlayer : MonoBehaviour
{
    [SerializeField]
    private PlayableCard PlayableCardPrefab;
    [SerializeField]
    private int scoreDiffToEndRound;
    private Player enemy;

    public void SetUp(Player humanPlayer)
    {
        enemy = humanPlayer;
    }

    public void ProcessTurn()
    {
        StartCoroutine(TurnRoutine());
    }

    private IEnumerator TurnRoutine()
    {
        yield return new WaitForSeconds(0.5f);

        if (GameManager.Instance.CurrentPlayer.CardsInHand.Count > 0
           && Mathf.Abs(GameManager.Instance.CurrentPlayer.Score - enemy.Score) <
scoreDiffToEndRound)
        {
            PlayCard();
        }
        else
        {
            EndRound();
        }
    }

    private void EndRound()
    {
        GameManager.Instance.EndRound();
    }

    private void PlayCard()
    {
        int index = Random.Range(0, GameManager.Instance.CurrentPlayer.CardsInHand.Count -
1);
        PlayableCard playableCard = Instantiate(PlayableCardPrefab, transform);
        playableCard.SetUp(GameManager.Instance.CurrentPlayer.CardsInHand[index],
Vector3.zero);
        playableCard.PlayCard();
    }
}
```

AudioManager.cs

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

public enum Sounds
{
    Button,
    Slider,
    CloseWindow,
    OpenWindow,
    SpawnCard,
    Buff,
    Debuff,
    Restore,
```

```
    Destroy,
    Victory,
    GameOver,
    EndRound
}

public enum Music
{
    MenuMusic,
    GameMusic
}

[Serializable]
public class SoundItem
{
    public Sounds Sound;
    public AudioClip AudioClip;
}

[Serializable]
public class MusicItem
{
    public Music Music;
    public AudioClip AudioClip;
}

public class AudioManager : MonoBehaviour
{
    public static AudioManager Instance;

    public bool IsMuted => isMuted;
    public string MusicKey = "MusicVolume";
    public string EffectsKey = "EffectsVolume";

    [SerializeField]
    private AudioMixer audioMixer;
    [SerializeField]
    private AudioSource musicSource;
    [SerializeField]
    private AudioSource soundsSource;
    [SerializeField]
    private List<SoundItem> soundsList;
    [SerializeField]
    private List<MusicItem> musicList;
    private bool isMuted;

    private void Start()
    {
        CreateInstance();
        SetUp();
    }

    private void CreateInstance()
    {
        Instance = this;
        DontDestroyOnLoad(this);
    }

    private void SetUp()
    {
        audioMixer.SetFloat(MusicKey, DataManager.Instance.SavedData.MusicVolume);
        audioMixer.SetFloat(EffectsKey, DataManager.Instance.SavedData.SoundVolume);
        isMuted = DataManager.Instance.SavedData.IsMuted;
        CheckIfMuted();
    }

    public void SetVolume(string key,float value)
    {
        float adjustedValue = Mathf.Log10(value) * 20;
        audioMixer.SetFloat(key, adjustedValue);
        DataManager.Instance.SaveAudioVolume(key, value);
    }

    public void SetMute(bool isMuted)
    {
        this.isMuted = isMuted;
     DataManager.Instance.SaveMutedStatus(isMuted);
        CheckIfMuted();
    }
```

```
        private void CheckIfMuted()
        {
            if (isMuted)
            {
                audioMixer.SetFloat(MusicKey, -80);
                audioMixer.SetFloat(EffectsKey, -80);
            }
            else
            {
                audioMixer.SetFloat(MusicKey, DataManager.Instance.SavedData.MusicVolume);
                audioMixer.SetFloat(EffectsKey, DataManager.Instance.SavedData.SoundVolume);
            }
        }

        public void PlaySound(Sounds sound)
        {
            soundsSource.PlayOneShot(GetSound(sound));
        }

        public void PlayMusic(Music music)
        {
            musicSource.clip = GetMusic(music);
            musicSource.Play();
        }

        private AudioClip GetSound(Sounds sound)
        {
            return soundsList.Find((value) => value.Sound == sound).AudioClip;
        }

        private AudioClip GetMusic(Music music)
        {
            return musicList.Find((value) => value.Music == music).AudioClip;
        }
    }
```

BaseCards.cs

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

public class BaseCard : MonoBehaviour
{
    [SerializeField]
    protected Transform characterContainer;
    [SerializeField]
    protected TextMeshProUGUI nameText;
    [SerializeField]
    protected TextMeshProUGUI strengthText;
    [SerializeField]
    protected TextMeshProUGUI descText;
    [SerializeField]
    protected Image rowImage;
    [SerializeField]
    protected Sprite firstRowSprite;
    [SerializeField]
    protected Sprite secondRowSprite;

    protected CardInfo cardInfo;

    protected virtual void SetTexts()
    {
        nameText.text = cardInfo.Type.ToString();
        strengthText.text = cardInfo.Strength.ToString();
        descText.text = cardInfo.Description;
    }

    protected void InitCharacter()
    {
        Transform character = Instantiate(cardInfo.CharacterPrefab,
characterContainer).transform;
        character.localPosition = cardInfo.CharacterPosition;
    }

    protected void SetFightingRowSprite()
    {
        rowImage.sprite = cardInfo.FightingRow == FightingRow.First?
            firstRowSprite :                secondRowSprite;
```

```
        }
    }

    CardForScan.cs
    using System;
    using System.Collections;
    using System.Collections.Generic;
    using TMPro;
    using UnityEngine;

    public class CardForScan : BaseCard
    {
        [SerializeField]
        private CardType type;
        [SerializeField]
        private DefaultObserverEventHandler eventHandler;

        private bool isInited;

        private void Start()
        {
            eventHandler.OnTargetFound.AddListener(TargetFound);
            eventHandler.OnTargetLost.AddListener(TargetLost);
        }

        private void TargetLost()
        {
        }

        private void TargetFound()
        {
            if (isInited)
            {
                return;
            }

            StatisticsManager.Instance.TrackEvent(TrackedEventType.ObjectRecognized,
type.ToString());
            AudioManager.Instance.PlaySound(Sounds.SpawnCard);
            cardInfo = CardsManager.Instance.CardInfos.Find((value) => value.Type == type);
            if (cardInfo != null)
            {
                isInited = true;
                SetTexts();
                InitCharacter();
                SetFightingRowSprite();

                DataManager.Instance.SaveCardIfNeeded(type);
            }
        }


    }

    CardsHolder.cs
    using System.Collections;
    using System.Collections.Generic;
    using UnityEngine;

    public class CardsHolder : MonoBehaviour
    {
        [SerializeField]
        private float border;
        [SerializeField]
        private Vector2 minMaxStep;
        [SerializeField]
        private Vector3 cardSelectedPosition;
        [SerializeField]
        private PlayableCard cardPrefab;
        [SerializeField]
        private Transform cardsParent;

        public void AddCard(CardInfo cardInfo)
        {
            PlayableCard spawnedCard = Instantiate(cardPrefab, cardsParent);
            spawnedCard.SetUp(cardInfo, cardSelectedPosition);
            SetUpCards();
        }

        public void SetUpCards()
```

```
    {
        float currentStep = (border * 2) / transform.childCount;
        currentStep = Mathf.Clamp(currentStep, minMaxStep.x, minMaxStep.y);

        Vector3 posiiton;
        Transform child;
        for (int i = 0; i < transform.childCount; i++)
        {
            child = transform.GetChild(i);
            posiiton = child.localPosition;
            posiiton.x = -border + i * currentStep;
            child.localPosition = posiiton;
        }
    }
}
```

CardsManager.cs

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

[System.Serializable]
public class CardInfo
{
    public CardType Type;
    public FightingRow FightingRow;
    public string Description;
    public int Strength;
    public CardPower CardPower;
    [Space]
    public Sprite imageSprite;
    public GameObject CharacterPrefab;
    public Vector3 CharacterPosition;
}

public enum CardType
{
    Astronaut,
    Drone,
    Oxygen,
    ElfWarrior,
    ElfArcher,
    ElfAssassin,
    ElfPriest,
    ElfQueen,

    HumanKnight,
    HumanArcher,
    HumanMage,
    HumanPaladin,
    HumanSoldier,

    OrcChieftan,
    OrcArcher,
    OrcGrunt,
    OrcShaman,
    OrcWarrior,

    UndeadAssassin,
    UndeadArcher,
    UndeadKing,
    UndeadMage,
    UndeadWarrior
}

public enum FightingRow
{
    First,
    Second
}

public enum CardPower
{
    None,
    Buff,
    Debuff,
    Restoration,
    DestroyCard
```

```
}

public class CardsManager : MonoBehaviour
{
    public static CardsManager Instance;

    public List<CardInfo> CardInfos = new List<CardInfo>();
    public List<CardInfo> UnlockedCards = new List<CardInfo>();

    private void Awake()
    {
        CreateInstance();
    }

    private void Start()
    {
        CollectUnlockedCards();
    }

    public void CollectUnlockedCards()
    {
        foreach (var cardType in DataManager.Instance.SavedData.SavedCardTypes)
        {
            UnlockedCards.Add(CardInfos.Find((value) => value.Type == cardType));
        }
    }

    private void CreateInstance()
    {
        Instance = this;
        DontDestroyOnLoad(this);
    }
}
```

CloseButton.cs

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

public class CloseButton : MonoBehaviour
{
    [SerializeField]
    private Scenes currentScene;

    void Start()
    {
        GetComponent<Button>().onClick.AddListener(Close);
    }

    private void Close()
    {
        AudioManager.Instance.PlaySound(Sounds.CloseWindow);
        SceneManager.LoadScene((int)Scenes.MainMenu);
        StatisticsManager.Instance.TrackEvent(TrackedEventType.TimeSpentInScene,
currentScene.ToString());
    }
}
```

CollectionCard.cs

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

public class CollectionCard : BaseCard
{
    [SerializeField]
    private Image cardImage;

    public void SetUp(CardInfo info)
    {
        cardInfo = info;
        cardImage.sprite = info.imageSprite;
        SetTexts();
        SetFightingRowSprite();
    }
}
```

CollectionsWindow.cs

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

public class CollectionsWindow : MonoBehaviour
{
    [SerializeField]
    private Transform cardsContainer;
    [SerializeField]
    private CollectionCard cardPrefab;

    void Start()
    {
        CardInfo cardInfo = null;

        foreach (var type in DataManager.Instance.SavedData.SavedCardTypes)
        {
            cardInfo = CardsManager.Instance.CardInfos.Find((value) => value.Type == type);
            if (cardInfo != null)
            {
                Instantiate(cardPrefab, cardsContainer).SetUp(cardInfo);
            }
        }
    }
}
```

DataManager.cs

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

[System.Serializable]
public class SavedData
{
    public List<CardType> SavedCardTypes;
    public float MusicVolume;
    public float SoundVolume;
    public bool IsMuted;
    public List<EventData> SavedEvents;
}

public class DataManager : MonoBehaviour
{
    public static DataManager Instance;
    public SavedData SavedData;

    private string savePath;

    private void Awake()
    {
        CreateInstance();
        savePath = Application.persistentDataPath + "/Save.txt";
        LoadSave();
    }

    private void CreateInstance()
    {
        Instance = this;
        DontDestroyOnLoad(this);
    }

    private void LoadSave()
    {
        string loadedData = string.Empty;

        StreamReader reader = new StreamReader(savePath);
        loadedData = reader.ReadToEnd();
        reader.Close();

        if (!string.IsNullOrEmpty(loadedData) && !string.IsNullOrWhiteSpace(loadedData))
        {
            JsonUtility.FromJsonOverwrite(loadedData, SavedData);
        }
    }

    public void SaveAudioVolume(string key, float value)
    {
```

```
        if (key == AudioManager.Instance.MusicKey)
        {
            SavedData.MusicVolume = value;
        }
        else
        {
            SavedData.SoundVolume = value;
        }
        Save();
    }

    public void SaveMutedStatus(bool isMuted)
    {
        SavedData.IsMuted = isMuted;
        Save();
    }

    public void SaveEvents(List<EventData> eventDatas)
    {
        SavedData.SavedEvents = eventDatas;
        Save();
    }

    private void Save()
    {
        string saveString = JsonUtility.ToJson(SavedData);

        StreamWriter writer = new StreamWriter(savePath, false);
        writer.WriteLine(saveString);
        writer.Close();
    }

    public void SaveCardIfNeeded(CardType type)
    {
        if (!SavedData.SavedCardTypes.Contains(type))
        {
            SavedData.SavedCardTypes.Add(type);
            Save();
        }
    }
}
```

EffectsManager.cs

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

public class EffectManager : MonoBehaviour
{
    public static EffectManager Instance;

    [SerializeField]
    private EffectPooler effectPooler;

    private void Awake()
    {
        Instance = this;
    }

    public void SpawnEffect(EffectType effectType, Transform container)
    {
        ParticleSystem particleSystem = effectPooler.GetEffect(effectType);
        SetUpParticleTransform(particleSystem.transform, container);
        particleSystem.gameObject.SetActive(true);
        particleSystem.Play();
    }

    private void SetUpParticleTransform(Transform particleTransform, Transform container)
    {
        particleTransform.parent = container;
        particleTransform.localPosition = Vector3.zero;
        particleTransform.localScale = Vector3.one;
    }

    public void ReturnToPool(GameObject returningObject)
    {
        effectPooler.ReturnToPool(returningObject);
    }
}
```

EffectObject.cs

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

public class EffectObject : MonoBehaviour
{
    [SerializeField]
    private float lifeTime;

    private void OnEnable()
    {
        StartCoroutine(LifeRoutine());
    }

    private IEnumerator LifeRoutine()
    {
        yield return new WaitForSeconds(lifeTime);
        EffectManager.Instance.ReturnToPool(gameObject);
    }
}
```

EffectPooler.cs

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

public enum EffectType
{
    Spawn,
    Buff,
    Debuff,
    Restore,
    Destroy
}

[Serializable]
public class Effect
{
    public EffectType EffectType;
    public ParticleSystem particlePrefab;
    public List<ParticleSystem> spawnedParticles = new List<ParticleSystem>();
}

public class EffectPooler : MonoBehaviour
{
    [SerializeField]
    private int startAmount;
    [SerializeField]
    private Transform effectsContainer;
    [SerializeField]
    private List<Effect> effects;

    private void Start()
    {
        PrepareParticles();
    }

    private void PrepareParticles()
    {
        foreach (var effect in effects)
        {
            for (int i = 0; i < startAmount; i++)
            {
                SpawnNewParticle(effect);
            }
        }
    }

    private ParticleSystem SpawnNewParticle(Effect effect)
    {
        ParticleSystem particle = Instantiate(effect.particlePrefab, effectsContainer);
        effect.spawnedParticles.Add(particle);
        particle.gameObject.SetActive(false);
        return particle;
    }

    public ParticleSystem GetEffect(EffectType type)
    {
```

```
                Effect selectedEffect = effects.Find((value) => value.EffectType == type);
                ParticleSystem particle = null;

                if (selectedEffect != null)
                {
                    particle = selectedEffect.spawnedParticles.Find((value) =>
value.gameObject.activeSelf == false);

                    if (particle == null)
                    {
                        particle = SpawnNewParticle(selectedEffect);
                    }
                }

                return particle;
            }

            public void ReturnToPool(GameObject returningObject)
            {
                returningObject.SetActive(false);
                returningObject.transform.parent = effectsContainer;
            }
        }

        GameManager.cs
        using System;
        using System.Collections;
        using System.Collections.Generic;
        using UnityEngine;

        [System.Serializable]
        public class Player
        {
            [HideInInspector]
            public bool IsInited;

            public bool IsHuman;
            public int Score;
            public int Victories;
            public List<GameRow> GameRows;
            public List<CardInfo> CardsInHand = new List<CardInfo>();
            public List<CardInfo> CardsInDeck = new List<CardInfo>();
        }

        public class GameManager : MonoBehaviour
        {
            public static GameManager Instance;
            public CardsHolder CardsHolder;
            public List<Player> players;
            [HideInInspector]
            public Player CurrentPlayer;

            [SerializeField]
            private int deckSize;
            [SerializeField]
            private int handSize;

            [SerializeField]
            private DefaultObserverEventHandler eventHandler;
            [SerializeField]
            private AIPlayer aiPlayer;
            [SerializeField]
            private GameUIManager uIManager;
            private int turn = -1;

            private void Start()
            {
                Instance = this;
                AudioManager.Instance.PlayMusic(Music.GameMusic);
                uIManager.ButtonPressed.AddListener(OnEndRoundButtonPressed);
                eventHandler.OnTargetFound.AddListener(StartGame);

        #if UNITY_EDITOR
                StartGame();
               // uIManager.ShowEndScreen(true);
        #endif
            }

            private void OnDestroy()
            {
```

```
    uIManager.ButtonPressed.RemoveListener(OnEndRoundButtonPressed);
    eventHandler.OnTargetFound.RemoveListener(StartGame);
}

private void OnEndRoundButtonPressed()
{
    if (CurrentPlayer.IsHuman)
    {
        EndRound();
    }
}

private void StartGame()
{
    aiPlayer.SetUp(players.Find((value) => value.IsHuman));
    NextTurn();
}

public void EndRound()
{
    AudioManager.Instance.PlaySound(Sounds.EndRound);

    int currentScore = 0;
    Player winner = null;

    foreach (var player in players)
    {
        if (player.Score > currentScore)
        {
            winner = player;
            currentScore = player.Score;
        }
    }

    winner.Victories++;

    if (winner.Victories >= 2)
    {
        uIManager.ShowEndScreen(winner.IsHuman);
    }
    else
    {
        NextTurn();
    }
}

private void NextTurn()
{
    SetCurrentPlayer();
    AddCardsToCurrentPlayer();

    if (!CurrentPlayer.IsHuman)
    {
        aiPlayer.ProcessTurn();
    }
}

private void SetCurrentPlayer()
{
    if (turn < players.Count - 1)
    {
        turn++;
    }
    else
    {
        turn = 0;
    }

    CurrentPlayer = players[turn];
}

private void AddCardsToCurrentPlayer()
{
    if (!CurrentPlayer.IsInited)
    {
        if (CardsManager.Instance.UnlockedCards.Count == 0)
        {
            CardsManager.Instance.CollectUnlockedCards();
        }
```

```
                    for (int i = 0; i < deckSize; i++)
                    {
                        int rand1 = UnityEngine.Random.Range(0,
CardsManager.Instance.CardInfos.Count);
                        CurrentPlayer.CardsInDeck.Add(CardsManager.Instance.CardInfos[rand1]);

                       // int rand = UnityEngine.Random.Range(0,
CardsManager.Instance.UnlockedCards.Count);
                       // CurrentPlayer.CardsInDeck.Add(CardsManager.Instance.UnlockedCards[rand]);
                    }

                    CurrentPlayer.IsInited = true;
                    for (int i = 0; i < handSize; i++)
                    {
                        AddCardToHand(CurrentPlayer.CardsInDeck[0]);
                    }
                }
                else
                {
                    if (CurrentPlayer.CardsInDeck.Count == 0)
                    {
                        return;
                    }

                    AddCardToHand(CurrentPlayer.CardsInDeck[0]);
                }
            }

            public void AddCardToHand(CardInfo cardInfo)
            {
                if (CurrentPlayer.IsHuman)
                {
                    CardsHolder.AddCard(cardInfo);
                }

                CurrentPlayer.CardsInHand.Add(cardInfo);
                CurrentPlayer.CardsInDeck.Remove(cardInfo);
            }

            public void RemoveCardFromHand(CardInfo cardInfo)
            {
                CurrentPlayer.CardsInHand.Remove(cardInfo);
            }

            public void PlaceCardInARow(PlayableCard playableCard)
            {
                if (CurrentPlayer.IsHuman)
                {
                    StatisticsManager.Instance.TrackEvent(TrackedEventType.CardPlayed,
playableCard.CardInfo.Type.ToString());
                }
                RemoveCardFromHand(playableCard.CardInfo);
                CurrentPlayer.GameRows.Find((value) => value.FightingRow ==
playableCard.CardInfo.FightingRow).PlaceCard(playableCard);
                NextTurn();
            }

            public void ApplyHostileEffect(Player attacker, FightingRow fightingRow, CardPower
cardPower)
            {
                Player target = players.Find((value) => value != attacker);
                GameRow row = target.GameRows.Find((value) => value.FightingRow == fightingRow);
                if (cardPower == CardPower.DestroyCard)
                {
                    row.DestroyCard();
                }

                else
                {
                   row.SetEffectToAllCards(CardState.Debuffed);
                }

            }
            public void UpdateScore(Player player, int amount)
            {
                player.Score += amount;
                uIManager.SetScoreText(player.IsHuman, player.Score);
            }
        }
```

GameRow.cs

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

public class GameRow : MonoBehaviour
{
    public FightingRow FightingRow;
    [SerializeField]
    private Transform content;
    [SerializeField]
    private List<PlayableCard> cards = new List<PlayableCard>();

    public void PlaceCard(PlayableCard card)
    {
        MoveToCardRow(card.transform);
        ApplyPower(card);
        cards.Add(card);
    }

    private void MoveToCardRow(Transform cardTransform)
    {
        cardTransform.parent = content;
        cardTransform.localScale = Vector3.one;
        cardTransform.localPosition = Vector3.zero;
        cardTransform.localRotation = Quaternion.identity;
    }

    public void ApplyPower(PlayableCard card)
    {
        switch (card.CardInfo.CardPower)
        {
            case CardPower.Buff:
                SetEffectToAllCards(CardState.Buffed);
                break;
            case CardPower.Debuff:
                GameManager.Instance.ApplyHostileEffect(card.Owner, FightingRow,
card.CardInfo.CardPower);
                break;
            case CardPower.Restoration:
                SetEffectToAllCards(CardState.Normal);
                break;
            case CardPower.DestroyCard:
                GameManager.Instance.ApplyHostileEffect(card.Owner, FightingRow,
card.CardInfo.CardPower);
                break;
        }
    }

    public void SetEffectToAllCards(CardState state)
    {
        foreach (var card in cards)
        {
            card.ApplyEffect(state);
        }
    }

    public void DestroyCard()
    {
        PlayableCard selectedCard = null;
        foreach (var card in cards)
        {
            if (selectedCard == null || card.CurrentStrength < selectedCard.CurrentStrength)
            {
                selectedCard = card;
            }
        }

        if (selectedCard)
        {
            selectedCard.DestroyCard();
        }
    }
}
```

GameUIManager.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using TMPro;
```

```
using UnityEngine;
using UnityEngine.Events;
using UnityEngine.UI;

public class GameUIManager : MonoBehaviour
{
    [HideInInspector]
    public UnityEvent ButtonPressed;

    [SerializeField]
    private VictoryScreen victoryScreen;
    [SerializeField]
    private Button endRoundButton;
    [SerializeField]
    private TextMeshProUGUI playerScore;
    [SerializeField]
    private TextMeshProUGUI enemyScore;

    void Start()
    {
        endRoundButton.onClick.AddListener(EndRound);
        SetScoreText(true, 0);
        SetScoreText(false, 0);
    }

    private void EndRound()
    {
        ButtonPressed.Invoke();
    }

    public void SetScoreText(bool isHuman, int score)
    {
        if (isHuman)
        {
            playerScore.text = "Player: " + score;
        }
        else
        {
            enemyScore.text = "Enemy: " + score;
        }
    }

    public void ShowEndScreen(bool isVictory)
    {
        victoryScreen.Show(isVictory);
    }
}
```

Loader.cs

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

public class Loader : MonoBehaviour
{
    void Start()
    {
        SceneManager.LoadScene((int)Scenes.MainMenu);
    }
}
```

ManagersHolder.cs

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

public class ManagersHolder : MonoBehaviour
{
    private void Awake()
    {
        DontDestroyOnLoad(this);
    }
}
```

MenuManager.cs

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

```
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public enum Scenes
{
    Preloader,
    MainMenu,
    Game,
    ScanCards,
    Settings,
    Collection
}

public class MenuManager : MonoBehaviour
{
    [SerializeField]
    private Button playButton;
    [SerializeField]
    private Button scanCardsButton;
    [SerializeField]
    private Button setttinsButton;
    [SerializeField]
    private Button collectionButton;
    [SerializeField]
    private Button exitButton;

    void Start()
    {
        StatisticsManager.Instance.TrackEvent(TrackedEventType.SceneEntered,
Scenes.MainMenu.ToString());
        AudioManager.Instance.PlayMusic(Music.MenuMusic);
        playButton.onClick.AddListener(Play);
        scanCardsButton.onClick.AddListener(ScanCards);
        setttinsButton.onClick.AddListener(Settings);
        collectionButton.onClick.AddListener(Collection);
        exitButton.onClick.AddListener(Exit);
    }

    private void Play()
    {
        LoadScene(Scenes.Game);
    }

    private void ScanCards()
    {
        LoadScene(Scenes.ScanCards);
    }

    private void Settings()
    {
        LoadScene(Scenes.Settings, LoadSceneMode.Additive);
    }

    private void Collection()
    {
        LoadScene(Scenes.Collection, LoadSceneMode.Additive);
    }

    private void Exit()
    {
        StatisticsManager.Instance.TrackEvent(TrackedEventType.TimeSpentInScene,
Scenes.MainMenu.ToString());
        AudioManager.Instance.PlaySound(Sounds.CloseWindow);
        Application.Quit();
    }

    private void LoadScene(Scenes scene, LoadSceneMode sceneMode = LoadSceneMode.Single)
    {
        StatisticsManager.Instance.TrackEvent(TrackedEventType.TimeSpentInScene,
Scenes.MainMenu.ToString());
        StatisticsManager.Instance.TrackEvent(TrackedEventType.SceneEntered,
scene.ToString());
        AudioManager.Instance.PlaySound(Sounds.Button);
        SceneManager.LoadScene((int)scene, sceneMode);
    }
}
```

PlayableCard.cs

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

```
using UnityEngine;
using UnityEngine.EventSystems;
using UnityEngine.UI;

public enum CardState
{
    Normal,
    Buffed,
    Debuffed
}

public class PlayableCard : BaseCard, IPointerDownHandler
{
    public CardInfo CardInfo => cardInfo;
    public CardState CardState => cardState;
    public Player Owner => owner;
    [HideInInspector]
    public int CurrentStrength;

    [SerializeField]
    private GameObject actionButtons;
    [SerializeField]
    private Button confirmButton;
    [SerializeField]
    private Button cancelButton;
    [SerializeField]
    private Image image;
    [SerializeField]
    private Transform effectContainer;

    private Player owner;
    private CardState cardState;
    private Vector3 normalPosition;
    private Vector3 selectedPosition;
    private bool isPlayed;
    private bool isSelected;

    private void Start()
    {
        confirmButton.onClick.AddListener(PlayCard);
        cancelButton.onClick.AddListener(Cancel);
    }

    public void OnPointerDown(PointerEventData eventData)
    {
        if (!isPlayed && !isSelected && GameManager.Instance.CurrentPlayer.IsHuman)
        {
            AudioManager.Instance.PlaySound(Sounds.Button);
            Select();
        }
    }

    private void Select()
    {
        isSelected = true;
        actionButtons.SetActive(true);
        normalPosition = transform.localPosition;
        transform.localPosition = selectedPosition;
    }

    public void SetUp(CardInfo info, Vector3 selectedPos)
    {
        selectedPosition = selectedPos;
        cardInfo = info;
        CurrentStrength = cardInfo.Strength;
        image.sprite = cardInfo.imageSprite;
        SetTexts();
        SetFightingRowSprite();
    }

    protected override void SetTexts()
    {
        nameText.text = cardInfo.Type.ToString();
        strengthText.text = CurrentStrength.ToString();
        descText.text = cardInfo.Description;
    }

    public void PlayCard()
    {
        actionButtons.SetActive(false);
```

```
        owner = GameManager.Instance.CurrentPlayer;
        isSelected = true;
        isPlayed = true;
        PutOnField();
    }

    public void ApplyEffect(CardState state)
    {
        if (cardState != state)
        {
            cardState = state;

            switch (state)
            {
                case CardState.Buffed:
                    AudioManager.Instance.PlaySound(Sounds.Buff);
                    EffectManager.Instance.SpawnEffect(EffectType.Buff, effectContainer);
                    UpdateStrength(1);
                    break;
                case CardState.Debuffed:
                    AudioManager.Instance.PlaySound(Sounds.Debuff);
                    EffectManager.Instance.SpawnEffect(EffectType.Debuff, effectContainer);
                    UpdateStrength(-1);
                    break;
                case CardState.Normal:
                    AudioManager.Instance.PlaySound(Sounds.Restore);
                    EffectManager.Instance.SpawnEffect(EffectType.Restore, effectContainer);
                    UpdateStrength(cardInfo.Strength - CurrentStrength);
                    break;
            }
        }
    }

    private void UpdateStrength(int amount)
    {
        CurrentStrength += amount;
        GameManager.Instance.UpdateScore(owner, amount);
        strengthText.text = CurrentStrength.ToString();
    }

    public void DestroyCard()
    {
        AudioManager.Instance.PlaySound(Sounds.Destroy);
        EffectManager.Instance.SpawnEffect(EffectType.Destroy, effectContainer);
        GameManager.Instance.UpdateScore(owner, -CurrentStrength);
        Destroy(gameObject,4f);
    }

    private void Cancel()
    {
        isSelected = false;
        actionButtons.SetActive(false);
        transform.localPosition = normalPosition;
        AudioManager.Instance.PlaySound(Sounds.CloseWindow);
    }

    private void PutOnField()
    {
        GameManager.Instance.PlaceCardInARow(this);
        GameManager.Instance.UpdateScore(owner, CurrentStrength);
        InitCharacter();
        EffectManager.Instance.SpawnEffect(EffectType.Spawn, effectContainer);
        AudioManager.Instance.PlaySound(Sounds.SpawnCard);
    }
}
```

SettingsWindow.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class SettingsWindow : MonoBehaviour
{
    [SerializeField]
    private Slider musicSlider;
    [SerializeField]
    private Slider effectsSlider;
    [SerializeField]
    private Toggle muteToggle;
    [SerializeField]
```

```
            private Button exitButton;

            public void Start()
            {
                AudioManager.Instance.PlaySound(Sounds.OpenWindow);
                exitButton.onClick.AddListener(CloseWindow);
                GetSettings();
                AddListeners();
            }

            private void CloseWindow()
            {
                StatisticsManager.Instance.TrackEvent(TrackedEventType.TimeSpentInScene,
Scenes.Settings.ToString());
                AudioManager.Instance.PlaySound(Sounds.CloseWindow);
                SceneManager.UnloadSceneAsync(gameObject.scene);
            }

            private void GetSettings()
            {
                musicSlider.value = DataManager.Instance.SavedData.MusicVolume;
                effectsSlider.value = DataManager.Instance.SavedData.SoundVolume;
                muteToggle.isOn = AudioManager.Instance.IsMuted;
            }

            private void AddListeners()
            {
                musicSlider.onValueChanged.AddListener(delegate { MusicVolumeChanged(); });
                effectsSlider.onValueChanged.AddListener(delegate { EffectsVolumeChanged(); });
                muteToggle.onValueChanged.AddListener(Mute);
            }

            private void MusicVolumeChanged()
            {
                AudioManager.Instance.SetVolume(AudioManager.Instance.MusicKey, musicSlider.value);
                AudioManager.Instance.PlaySound(Sounds.Slider);
            }

            private void EffectsVolumeChanged()
            {
                AudioManager.Instance.SetVolume(AudioManager.Instance.EffectsKey,
effectsSlider.value);
                AudioManager.Instance.PlaySound(Sounds.Slider);
            }

            private void Mute(bool isMute)
            {
                AudioManager.Instance.SetMute(isMute);
            }
        }

        StatisticsManager.cs
        using System;
        using System.Collections;
        using System.Collections.Generic;
        using UnityEngine;

        [System.Serializable]
        public class EventData
        {
            public string Id;
            public double result;

            public EventData(string id, double data)
            {
                Id = id;
                result = data;
            }
        }

        public enum TrackedEventType
        {
            SceneEntered,
            TimeSpentInScene,
            ObjectRecognized,
            CardPlayed,
            GameEnd
        }
```

```
        public class StatisticsManager : MonoBehaviour
        {
            public static StatisticsManager Instance;
            public string VictoryKey = "Victory";
            public string DefeatKey = "Defeat";

            private List<EventData> eventDatas = new List<EventData>();
            private DateTime sceneEnteredTime;
            private DateTime lastRecognitionTime;

            void Awake()
            {
                CreateInstance();
            }

            private void Start()
            {
                eventDatas = DataManager.Instance.SavedData.SavedEvents;
            }

            private void CreateInstance()
            {
                Instance = this;
                DontDestroyOnLoad(this);
            }

            public void TrackEvent(TrackedEventType type, string content)
            {
                switch (type)
                {
                    case TrackedEventType.SceneEntered:
                        TrackSceneEntered(content);
                        break;
                    case TrackedEventType.TimeSpentInScene:
                        TrackTimeSpentInScene(content);
                        break;
                    case TrackedEventType.ObjectRecognized:
                        TrackRecognitionTime(content);
                        break;
                    case TrackedEventType.CardPlayed:
                        TrackCardPlayed(content);
                        break;
                    case TrackedEventType.GameEnd:
                        TrackGameEndEvent(content);
                        break;
                }

                DataManager.Instance.SaveEvents(eventDatas);
            }

            private void TrackSceneEntered(string sceneName)
            {
                GetEventData(sceneName).result++;
                sceneEnteredTime = DateTime.Now;
            }

            private void TrackTimeSpentInScene(string sceneName)
            {
                GetEventData("Time spent in " +  sceneName).result +=
GetTotalSeconds(sceneEnteredTime);
            }

            private double GetTotalSeconds(DateTime start)
            {
                return (DateTime.Now - start).TotalSeconds;
            }

            private void TrackRecognitionTime(string objectName)
            {
                double result = GetTotalSeconds(lastRecognitionTime);
                EventData eventData = GetEventData(objectName + "recognized in ");
                if (eventData.result == 0 || eventData.result > result)
                {
                    eventData.result = result;
                }

                TrackLastRecognitionTime();
            }

            public void                                TrackLastRecognitionTime()
```

```
            {
                lastRecognitionTime = DateTime.Now;
            }

            private void TrackCardPlayed(string cardId)
            {
                GetEventData(cardId + " played").result++;
            }

            private void TrackGameEndEvent(string endGameResult)
            {
                EventData victories = GetEventData("victories");
                EventData defeats = GetEventData("defeats");

                if (endGameResult == VictoryKey)
                {
                    victories.result++;
                }
                else
                {
                    defeats.result++;
                }

                GetEventData("WinRatio").result = victories.result / defeats.result;
            }

            private EventData GetEventData(string eventId)
            {
                EventData eventData = eventDatas.Find((value) => value.Id == eventId);

                if (eventData == null)
                {
                    eventData = new EventData(eventId, 0);
                    eventDatas.Add(eventData);
                }

                return eventData;
            }
        }
```

VictoryScreen.cs

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

public class VictoryScreen : MonoBehaviour
{
    [SerializeField]
    private Button retryButton;
    [SerializeField]
    private Button menuButton;
    [SerializeField]
    private Text headerText;
    [SerializeField]
    private Text mainText;

    private void Start()
    {
        retryButton.onClick.AddListener(Retry);
        menuButton.onClick.AddListener(LoadMenu);
    }

    private void Retry()
    {
        AudioManager.Instance.PlaySound(Sounds.Button);
        SceneManager.LoadScene((int)Scenes.Game);
    }

    private void LoadMenu()
    {
        StatisticsManager.Instance.TrackEvent(TrackedEventType.TimeSpentInScene,
Scenes.Game.ToString());
        AudioManager.Instance.PlaySound(Sounds.Button);
        SceneManager.LoadScene((int)Scenes.MainMenu);
    }

    public void Show(bool isVictory)
```

```
            {
                if (!isVictory)
                {
                    StatisticsManager.Instance.TrackEvent(TrackedEventType.GameEnd,
StatisticsManager.Instance.DefeatKey);
                    AudioManager.Instance.PlaySound(Sounds.GameOver);
                    headerText.text = "Defeat";
                    mainText.text = "Game Over!";
                }
                else
                {
                    StatisticsManager.Instance.TrackEvent(TrackedEventType.GameEnd,
StatisticsManager.Instance.VictoryKey);
                    AudioManager.Instance.PlaySound(Sounds.Victory);
                }

                gameObject.SetActive(true);
            }
        }
```

Додаток Б

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ДНІПРОВСЬКА ПОЛІТЕХНІКА»

Факультет інформаційних технологій
Кафедра програмного забезпечення комп'ютерних систем

ВІДГУК

Наукового керівника Приходченко С. Д., доцента, кафедри ПЗКС
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання, посада, місце роботи)

На кваліфікаційну роботу
студента Тріска Антона Євгеновича
(прізвище, ім'я, по батькові)

курсу II групи 122м-22-2
спеціальності 122 Комп'ютерні науки
на тему Дослідження способів та принципів взаємодії з Artificial Reality на прикладі ігрового додатку

Актуальність теми Актуальність теми обумовлена стрімким розвитком технологій в галузі Artificial Reality (AR) тазростаючим інтересом до ігрових додатків, яківикористовують AR.

Мета досліджень підвищення ефективності використання AR технологій в ігрових додатках.

Коротка характеристика розділів роботи У першому розділі проведено аналіз предметної області, де досліджується потенціал використання AR в ігровій індустрії. У другому розділі розглянуто методи та технології, які можуть бути використані для оптимізації взаємодії гравця з AR-ігровим додатком. Третій розділ описує розроблений ігровий додаток, що використовує додану реальність та аналізується результати його роботи.

Практичне значення роботи Полягає в вдосконалені використання AR технологій в ігрових додатках шляхом виділення найбільш підходящих для

конкретних цілей в умовах обмеженого часу та бюджету розробки.

Зауваження та недоліки Наявні певні недоліки в оформлені роботи, частина інформації розкрита недостатньо

Висновки та оцінка Кваліфікаційна робота заслуговує оцінки «добре», виконавець заслуговує на присвоєння відповідної кваліфікації.

Науковий керівник Приходченко Сергій Дмитрович, професор, каф. ПЗКС
(прізвище, ім'я, по батькові, посада, місце роботи)

«_13_»___грудня___2022 р. _____
(підпис)

Додаток В

РЕЦЕНЗІЯ

на кваліфікаційну роботу

студента Тріска Антона Євгеновича
(прізвище, ім'я, по батькові)
курсу II групи 122м-22-2
кафедри програмного забезпечення комп'ютерних систем
спеціальності 122 Комп'ютерні науки
Тема роботи Дослідження способів та принципів взаємодії з Artificial Reality на прикладі ігрового додатку

Стисла характеристика розділів роботи Перший розділ надає інформацію про предметну область та її поточний стан. В другому розділі досліджено та описано існуючі на даний момент способи та принципи взаємодії з доповненою реальністю. Третій розділ містить інформацію про розроблене програмне забезпечення, його функціонал та проаналізовано результати роботи.

Пропозиції, внесені студентом, рівень їх наукового обґрунтування
Дана робота, написана згідно стандартів, використовуючи корректну термінологію. Висновки з роботи є адекватними, виведеними з дослідження проведеного протягом її написання

Практичне значення роботи полягає в досліжені та аналізі наявних способів я та принципів взаємодії з AR, що дозволить використати цю інформацію для більш швидкого та зручного створення програмного забезпечення з використанням доданої реальності.

Якість оформлення роботи робота виконана у відповідності до вимог оформлення кваліфікаційних робіт магістерського рівня і відповідає поставленій задачі.

Недоліки в роботі у роботі недостатньо розкриті глибинні принципи, що відповідають за спосіб роботи кожної конкретної технології AR.

Загальний висновок отримані результати є закінченою науково-дослідною роботою і мають практичну цінність, що підтверджує здатність Тріска Антона Євгеновича до самостійного ведення наукової роботи та вміння з проектування та розробки програмного забезпечення. Кваліфікаційна робота заслуговує оцінки "добре", а Тріско А. Є. – присвоєння відповідної кваліфікації.

(підготовленість студента до самостійної роботи як спеціаліста)

Оцінка магістерської роботи "добре"

Рецензент _____

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

«___»_____20__ р. _____

(підпис)

ДОДАТОК Г

ПЕРЕЛІК ДОКУМЕНТІВ НА ОПТИЧНОМУ НОСІЇ

Ім’я файла	**Опис**
Пояснювальні документи	
Кваліфікаційна робота Тріско.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна робота Тріско.pdf	Пояснювальна записка роботи в форматі PDF
Програма	
Program.zip	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація Тріско.pptx	Презентація кваліфікаційної роботи