Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження особливостей використання нативних AR інструментів в iOS при розробці програмного забезпечення
(тема)

Виконав:
Здобувач 2 року навчання
групи ІПЗм-23-1

Дар'я ПЕРВЄЄВА
(Власне ім'я, ПРІЗВИЩЕ)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-наукова

Керівник доц. Олена ШЕВЧЕНКО
(посада, Власне ім'я, ПРІЗВИЩЕ)

Допускається до захисту
Зав. кафедри _____ Кирило СМЕЛЯКОВ
(підпис) (Власне ім'я, ПРІЗВИЩЕ)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти другий (магістерський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми освітньо-наукова програма
Освітня програма Інженерія програмного забезпечення
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Первєєвій Дар’ї Олександрівні
(прізвище, ім’я, по батькові)

1. Тема роботи «Дослідження особливостей використання нативних AR інструментів в iOS при розробці програмного забезпечення»
Затверджена наказом по університету від 15.04. 2025р. № 290 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 18.06.2025
3. Вихідні дані до роботи електронні ресурси за обраною тематикою, специфікації фреймворків ARKit, SceneKit та RealityKit, вимоги до реалізації сценаріїв тестування продуктивності, середовище розробки Xcode, мова програмування Swift, пристрій iPhone з підтримкою AR, документація Apple Developer.
4. Перелік питань, що потрібно опрацювати в роботі
вступ, аналіз предметної області, огляд літературних джерел, розробка сценаріїв тестування продуктивності, аналіз особливостей роботи фреймворків ARKit, SceneKit та RealityKit, визначення критеріїв порівняння, нормалізація критеріїв, порівняння їх продуктивності, опис отриманих результатів, рекомендації для вибору фреймворків залежно від потреб проекту, застосування багатокритеріального аналізу, висновки, перелік джерел посилань, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання	16.04.2025	Виконано
2	Аналіз предметної галузі і постановка задачі	18.04.2025	Виконано
3	Визначення та аналіз додаткових критеріїв порівняння фреймворків, які впливають на користувацький досвід	22.04.2025	Виконано
4	Формалізація критеріїв	24.04.2025	Виконано
5	Проектування та розробка програмного забезпечення	30.04.2025	Виконано
6	Вимірювання продуктивності	02.05.2025	Виконано
7	Підготовка до апробації результатів дослідження. Публікація матеріалів	10.05.2025	Виконано
8	Підготовка пояснювальної записки	20.05.2025	Виконано
9	Підготовка презентації та доповіді	31.05.2025	Виконано
10	Перевірка на плагіат	08.06.2025	Виконано
11	Нормоконтроль	10.06.2025	Виконано
12	Рецензування	12.06.2025	Виконано
13	Попередній захист	14.06.2025	Виконано
14	Занесення диплома в електронний архів	16.06.2025	Виконано
15	Допуск до захисту у зав. кафедри	18.06.2025	Виконано

Дата видачі завдання 16 квітня 2025р.

Здобувач _____ Дар’я ПЕРВЄЄВА
(підпис)

Керівник роботи _____ доц. Олена ШЕВЧЕНКО
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка містить: 106 стор., 12 рис., 11 табл., 20 джерел.

ПРОДУКТИВНІСТЬ, ТЕСТУВАННЯ, РЕНДЕРИНГ, ARKIT, CPU, FPS, GPU, IOS, REALITY KIT, SCENEKIT, SWIFT, XCODE INSTRUMENTS.

Об’єктом дослідження є нативні фреймворки, які використовуються для розробки AR-додатківна для iOS.

Предметом дослідження виступають продуктивність, архітектурні особливості, функціональні можливості, обмеження та юзабіліті фреймворків.

Метою роботи є виділення, формалізація та оцінка критеріїв, необхідних для обрання фреймворку, релевантного розроблюваному проекту, на основі багатокритеріального аналізу.

Методами дослідження є аналіз проблемної області, багатокритеріальний аналіз, проведення експерименту для виміру обраних показників.

У результаті роботи було досліджено фреймворки SceneKit та RealityKit, визначено критерії їх порівняння, експертно оцінені якісні критерії та спроектовано, реалізовано програмний додаток та проведено експеримент для оцінки кількісних критеріїв. Це дає можливість надавати обґрунтовані рекомендації щодо застосування конкретного фреймворку залежно від вимог проекту.

ARKIT, CPU, FPS, GPU, IOS, PERFORMANCE, REALITY KIT, RENDERING, SCENEKIT, SWIFT, TESTING, XCODE INSTRUMENTS

The object of the study is native frameworks used for the development of AR applications for iOS.

The subject of the study includes the performance, architectural features, functionality, limitations, and usability of the frameworks.

The aim of this work is to identify, formalize, and evaluate criteria necessary for selecting a framework relevant to the project being developed, based on multi-criteria analysis.

The research methods include analysis of the problem domain, multicriteria analysis, and experimental measurements of selected indicators.

As a result of the work, the SceneKit and RealityKit frameworks were studied, comparison criteria were defined, qualitative criteria were evaluated by experts, and a software application was designed and implemented to measure quantitative indicators. This enables well-grounded recommendations for selecting an appropriate framework depending on the project requirements.

Завідувачу кафедри ПІ
проф. Кирилу СМЕЛЯКОВУ

ЗАЯВА

щодо самостійності виконання кваліфікаційної роботи та можливості її публікації (та/або публікації анотації кваліфікаційної роботи) в електронному архіві відкритого доступу ElAr KhNURE

Я, Первєєва Дар'я Олександрівна, здобувач вищої освіти на другому (магістерському) рівні вищої освіти академічної групи ІПЗм-23-1 кафедра програмної інженерії, заявляю: моя кваліфікаційна робота на тему «Дослідження особливостей використання нативних AR інструментів в iOS при розробці програмного забезпечення», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в репозиторії "ElArKhNURE". Погоджуюся з авторським договором, відповідно до Положення про репозиторій ХНУРЕ "ElArKhNURE". Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлена з вимогами академічної доброчесності, згідно з якими виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

Дата 07.06.2025 Підпис 

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

AHP – Analytic Hierarchy Process (аналітичний ієрархічний процес).

AR – Augmented Reality (доповнена реальність).

ARKit – Framework для створення додатків доповненої реальності на платформі iOS.

CPU – Central Processing Unit (центральний процесор).

FPS – Frames Per Second (кадри на секунду).

GPU – Graphics Processing Unit (графічний процесор).

iOS – Operating System (операційна система) для мобільних пристроїв Apple.

LOD – Level Of Detail (рівень деталізації).

MCDA – Multi-Criteria Decision Analysis (багатокритеріальний аналіз).

PBR – Physical Based Rendering (рендерінг, заснований на фізиці).

VR – Virtual reality (віртуальна реальність).

ВСТУП

У сучасному світі інформаційних технологій зростає попит на розробку ефективних і стабільних AR-додатків. Віртуальна та доповнена реальність дедалі активніше інтегрується в різні сфери, включаючи освіту, розваги, медицину, архітектуру та електронну комерцію. Це обумовлює необхідність створення AR-додатків, що відповідають високим стандартам продуктивності та забезпечують ефективне використання апаратних ресурсів [1].

Фреймворки ARKit, SceneKit і RealityKit, розроблені компанією Apple, є основними інструментами для створення AR-додатків на платформі iOS. ARKit забезпечує стабільну інтеграцію віртуальних об'єктів у реальний світ, SceneKit – якісний рендеринг 3D-графіки, тоді як RealityKit дозволяє створювати візуалізації з високою деталізацією та фотореалістичними ефектами. Водночас розробники стикаються з необхідністю аналізу продуктивності цих фреймворків, щоб обирати оптимальні рішення залежно від потреб конкретного проєкту.

Актуальність теми дослідження полягає у важливості оптимізації використання апаратних ресурсів, таких як CPU та GPU, під час розробки AR-додатків. Обмежені можливості мобільних пристроїв часто створюють виклики для розробників, знижуючи продуктивність додатків і погіршуючи користувацький досвід. Дослідження ефективності роботи фреймворків є важливим етапом для вирішення цих проблем.

Мета роботи – дослідження впливу фреймворків на продуктивність та функціональність розроблюваних сцен у AR-додатках на iOS.

Завдання дослідження:

– проаналізувати літературні джерела для виявлення існуючих підходів до порівняння та аналізу фреймворків;
– визначити критерії для оцінки продуктивності та функціональності фреймворків, включаючи їх обмеження, типи критеріїв і шкали вимірювання;
– розробити програму для проведення експериментів і порівняння фреймворків за заданими критеріями.

Об’єкт дослідження – нативні фреймворки, які використовуються для розробки AR-додатківна для iOS.

Предмет дослідження – продуктивність, архітектурні особливості, функціональні можливості, обмеження та юзабіліті фреймворків.

Методи дослідження – аналіз проблемної області, багатокритеріальний аналіз (MCDA), проведення експерименту для виміру обраних показників.

Наукова новизна полягає у використанні MCDA для порівняння AR-фреймворків під iOS для надання рекомендацій щодо їх використання у розробці графічних сцен.

Практична значущість полягає у тому, що результати дослідження та розроблені рекомендації можуть бути використані розробниками для оцінки та оптимізації продуктивності AR-додатків, що дозволить покращити якість продуктів та ефективність використання апаратних ресурсів. Результати дослідження сприятимуть підвищенню стабільності та продуктивності AR-додатків.

Результати роботи були апробовані на XVIII Міжнародній науково-практичній конференції магістрантів та аспірантів «Теоретичні та практичні дослідження молодих вчених» [2].

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз проблемної області дослідження

У сучасному світі доповнена реальність (AR) та віртуальна реальність (VR) є одними з найбільш перспективних напрямів розвитку інформаційних технологій. AR-додатки інтегруються у різноманітні сфери, зокрема освіту, медицину, архітектуру, розваги, електронну комерцію та промисловість. Це зумовлено можливістю AR забезпечувати взаємодію користувача з цифровим контентом у реальному світі, що створює новий рівень інтерактивності та персоналізації.

Популяризація AR-додатків сприяла зростанню кількості програмних інструментів для їх розробки. На платформі iOS ключовими рішеннями є фреймворки ARKit у комбінації з SceneKit та RealityKit, кожен з яких надає набір інструментів для роботи з AR. ARKit виступає базовою платформою для інтеграції AR у мобільні пристрої, SceneKit фокусується на створенні тривимірної графіки, а RealityKit забезпечує високоякісну візуалізацію з використанням сучасних графічних технологій, таких як фізично коректне освітлення та тіньові ефекти.

Попри широкі можливості цих інструментів, існує низка проблем, які обмежують їх використання. AR-додатки потребують значних обчислювальних ресурсів для обробки тривимірної графіки, трекінгу об’єктів та рендерингу. Високе навантаження на CPU та GPU може призводити до падіння FPS (кількості кадрів за секунду) та погіршення користувацького досвіду. Під час роботи з AR-додатками можуть виникати проблеми з трекінгом, втратами продуктивності при збільшенні кількості об’єктів у сцені, а також збої в обробці даних. Вибір фреймворку та підхід до розробки істотно впливають на швидкодію додатка. Розробники стикаються з проблемою пошуку балансу між продуктивністю, якістю графіки та складністю реалізації.

Для вирішення зазначених проблем необхідно проводити глибокий аналіз можливостей кожного фреймворку та розробляти стратегії їх ефективного використання. У цьому контексті актуальним є дослідження таких параметрів, як завантаження CPU та GPU, кількість кадрів за секунду (FPS), енергоспоживання та стабільність роботи додатків.

Наукова спільнота та індустрія пропонують низку підходів до аналізу продуктивності фреймворків. Найбільш поширеними є:

– тестування продуктивності: використання реальних сценаріїв для оцінки FPS, часу рендерингу та затримок у відображенні;
– профілювання ресурсів: застосування інструментів, таких як Xcode Instruments, для аналізу навантаження на процесор та відеокарту;
– симуляція складних умов: створення сцен з великою кількістю об’єктів або складною фізикою для оцінки стійкості фреймворків до екстремальних навантажень.

Варто зазначити, що продуктивність фреймворків залежить не лише від їх архітектури, але й від підходу розробника до проєктування сцен, оптимізації геометрії, текстур і коду. Наприклад, SceneKit дозволяє створювати кастомні 3D-об’єкти, але потребує більше зусиль для оптимізації порівняно з RealityKit, який має вбудовані засоби для спрощення цих задач.

Таким чином, аналіз проблемної області вказує на необхідність дослідження продуктивності ARKit, SceneKit та RealityKit з урахуванням реальних умов використання. Результати такого дослідження допоможуть визначити оптимальний фреймворк для розробки AR-додатків залежно від потреб проєкту, що забезпечить стабільність роботи, високу продуктивність та ефективне використання ресурсів мобільних пристроїв.

1.2 Аналіз існуючих методів реалізації та їх обмежень

На платформі iOS основними фреймворками для роботи з AR є ARKit у комбінації з SceneKit і RealityKit. Кожен із цих інструментів має як переваги, так і обмеження, що впливають на вибір підходу до їх застосування.

ARKit є базовим інструментом, який відповідає за розпізнавання реального середовища, трекінг камер, відстеження поверхонь та освітлення. Завдяки ARKit розробники отримують доступ до широких можливостей щодо інтеграції віртуальних об’єктів у реальний світ із врахуванням просторових характеристик.

Основні підходи до роботи з ARKit включають:

– використання базових функцій трекінгу для створення інтерактивних додатків;
– поєднання ARKit із іншими фреймворками, такими як SceneKit і RealityKit, для покращення графіки та взаємодії;
– SceneKit є інструментом для створення 3D-сцен із використанням текстур, матеріалів, світла та анімацій.

Основні переваги SceneKit:

– гнучкість у створенні кастомних тривимірних об’єктів;
– широкі можливості для програмної взаємодії зі сценами.

Обмеження SceneKit полягають у тому, що цей фреймворк не оптимізований для роботи з ARKit на рівні продуктивності, особливо в складних сценах із великою кількістю об’єктів.

RealityKit – сучасний фреймворк, який дозволяє створювати реалістичні сцени завдяки вбудованій фізиці, розширеній підтримці матеріалів та інтеграції з ARKit. Особливості підходу до використання RealityKit:

– інтуїтивний API, що спрощує розробку;
– висока якість візуалізації завдяки фізично коректному рендерингу.

Недоліком RealityKit є його обмежена кастомізація у порівнянні зі SceneKit.

Існуючі інструменти мають певні обмеження. Мобільні пристрої мають обмежені ресурси CPU, GPU та пам’яті, що може призводити до зниження продуктивності додатків, особливо при обробці складних сцен. Залежно від складності сцени, використання фреймворків може призводити до падіння FPS, що негативно впливає на користувацький досвід.

Розробники обирають фреймворки залежно від специфіки проєкту. Для простих AR-додатків із базовою інтерактивністю найчастіше використовують ARKit у поєднанні з SceneKit, що дозволяє швидко створювати високоякісні рішення. Для більш складних проєктів, де потрібна широка кастомізація, розробники звертаються до RealityKit, попри його вищі вимоги до оптимізації.

Незважаючи на широкі можливості, обмеження фреймворків вимагають подальших досліджень для розробки стратегій їх оптимального використання. Це

включає аналіз продуктивності у складних умовах, таких як збільшення кількості об’єктів або застосування складних фізичних ефектів. Успішне вирішення цих проблем дозволить розробникам створювати стабільні, продуктивні та якісні AR-додатки, які задовольняють сучасні вимоги користувачів.

1.3 Сучасні тенденції розвитку та перспективи доповненої реальності

Сучасний етап розвитку доповненої реальності (AR) визначається інтеграцією цієї технології у численні сфери діяльності людини. Використання AR активно зростає в освіті, медицині, архітектурі, маркетингу, розвагах та електронній комерції. Тенденції розвитку AR обумовлені вдосконаленням апаратного забезпечення, зокрема мобільних пристроїв, та еволюцією програмного забезпечення. Це створює нові можливості для розробників, але одночасно вимагає вирішення низки технічних і методологічних завдань.

Однією з ключових тенденцій є підвищення продуктивності мобільних пристроїв. Вбудовані процесори з високою обчислювальною потужністю, графічні ядра та сенсори, зокрема LiDAR, дозволяють покращити якість трекінгу середовища та рендерингу AR-сцен. Це відкриває можливості для створення більш реалістичних і функціональних додатків. Водночас зростає потреба в оптимізації використання обчислювальних ресурсів, адже складні сцени та динамічні об'єкти можуть перевантажувати процесор і графічний адаптер, що негативно впливає на стабільність роботи додатків.

Перспективним напрямом розвитку AR є автоматизація процесів розробки AR-додатків. Нові інструменти та бібліотеки дозволяють зменшити складність реалізації AR-рішень, забезпечуючи швидкий доступ до базових функціональних можливостей.

2 ОГЛЯД Й АНАЛІЗ ЛІТЕРАТУРНИХ, НАУКОВИХ ДЖЕРЕЛ

2.1 Огляд літературних джерел

У рамках дослідження продуктивності та ефективності фреймворків ARKit, SceneKit і RealityKit було проведено комплексний огляд літератури й наукових джерел для виявлення сучасних підходів і технологій у галузі розробки AR-додатків, аналізу продуктивності, а також оптимізації використання апаратних ресурсів. Відбір джерел здійснювався за чітко визначеними критеріями, які забезпечували високу якість і актуальність інформації для поставлених завдань дослідження.

Насамперед, використовувалися наукові статті й публікації в рецензованих журналах, представлені провідними дослідниками в галузі AR та комп'ютерної графіки. Особливу увагу приділено роботам, опублікованим на міжнародних конференціях і в журналах, таких як IEEE Xplore, ACM Digital Library та інші. Це забезпечило високу достовірність представлених даних. Крім того, значну роль відіграла офіційна документація Apple щодо ARKit, SceneKit і RealityKit, яка містить детальну інформацію про функціональні можливості та обмеження фреймворків.

Актуальність літературних джерел забезпечувалася відбором матеріалів за останні 5-10 років, щоб охопити найсучасніші інновації та підходи до створення AR-додатків і оптимізації рендерингу. Це дозволило відобразити тенденції у використанні фреймворків та технологій для підвищення продуктивності AR-додатків.

Об'єктивність і достовірність інформації оцінювалися за методологією досліджень у вибраних джерелах. Аналіз результатів різних робіт допоміг виявити ключові тенденції та порівняти підходи до оптимізації рендерингу й управління апаратними ресурсами, такими як CPU та GPU. Групування джерел за темами забезпечило структурований підхід до аналізу матеріалу. Було виділено чотири основні категорії:

– особливості фреймворків ARKit, SceneKit і RealityKit;
– методи оцінки продуктивності;

– оптимізація рендерингу в AR-додатках;
– інструменти тестування, такі як Xcode Instruments.

Для пошуку літератури використовувалися такі наукові бази даних, як IEEE Xplore, Google Scholar, а також офіційна документація Apple і технічні блоги. Це забезпечило широкий спектр джерел, включаючи наукові статті, технічні звіти та практичний досвід.

Чіткі критерії включення джерел гарантували їх актуальність і наукову цінність. До таких критеріїв належали: відповідність темі, висока якість методології та значимість внеску в обрану галузь. Відбір джерел також передбачав виключення застарілої або низькоякісної інформації. Синтез даних з різних джерел дозволив отримати об’єктивну картину поточного стану використання ARKit, SceneKit і RealityKit, а також визначити напрями для подальших досліджень.

Огляд літератури створив основу для аналізу фреймворків і розробки методології тестування продуктивності. Систематичний підхід до збору та аналізу даних допоміг створити обґрунтовану базу для подальшого дослідження, що дозволить запропонувати оптимальні рішення для розробників AR-додатків на iOS.

2.2 Аналіз літератури

Аналіз літератури є важливим етапом дослідження, оскільки дозволяє глибше зрозуміти існуючі підходи до використання фреймворків ARKit, SceneKit і RealityKit, оцінити їхні переваги та недоліки, а також виявити прогалини, які можуть стати основою для подальших досліджень. У цьому розділі розглядаються основні дослідження, опубліковані з 2021 року, які висвітлюють різні аспекти продуктивності, рендерингу та оптимізації роботи з AR-додатками, а також порівняння ефективності фреймворків у реальних умовах використання.

Розвиток технологій доповненої реальності (AR) на платформі iOS активно обговорюється в наукових і технічних статтях. Однією з ключових бібліотек є ARKit, представлена Apple у 2017 році. Багато досліджень розглядають можливості цього інструменту, а також його порівняння з іншими платформами.

У статті "Capabilities of ARCore and ARKit Platforms for AR/VR Applications" [2] порівнюються ARCore (Google) та ARKit (Apple), аналізуючи продуктивність за критеріями, як-от використання процесора, пам'яті, виявлення площин і робота в умовах низької освітленості. Результати тестів дозволяють визначити, яка платформа більше підходить для певних завдань.

У статті "Motion Tracking in iOS Applications Using Augmented Reality" [4] розглядається використання ARKit для точного відстеження руху користувачів на основі даних з камери, акселерометра, гіроскопа та магнітометра. Це забезпечує точне позиційне відстеження, необхідне для реалістичних AR-додатків.

"Towards Preventing Gaps in Health Care Systems Through Smartphone Use" [5] досліджує можливість використання ARKit для точного вимірювання відстаней у медичних додатках. Точність вимірювань залежить від кута огляду та моделі пристрою, що підтверджує потенціал ARKit у розробці медичних додатків.

SceneKit відома своїми можливостями для рендерингу 3D-графіки. У книзі "Building Your First ARKit App with SceneKit" [6] описано інтеграцію SceneKit для створення 3D-контенту. SceneKit дозволяє імпортувати 3D-об'єкти та працювати з анімаціями, фізичною симуляцією та ефектами частинок, що робить його корисним для дизайну, ігор і розваг.

RealityKit у статті "Augmented Reality SDK Overview for General Application Use" [7] представлено як інструмент із фотореалістичним рендерингом, фізичними симуляціями та просторовим звуком. RealityKit спрощує інтеграцію з ARKit і дозволяє створювати реалістичні AR-сцени, але має певні обмеження, як-от відсутність підтримки шейдерів.

У статті "Comparison of ARKit, ARCore and supported rendering technologies in iOS development" [8] RealityKit описано як фреймворк для ефективного рендерингу та роботи з фізичними взаємодіями. SceneKit залишається популярним для рендерингу 3D-графіки, але менш ефективний у сучасних AR-додатках.

Аналіз джерел свідчить, що ARKit і RealityKit є найсучаснішими інструментами для AR-додатків. SceneKit залишається актуальним для 3D-графіки, але потребує перевірки продуктивності у порівнянні з іншими фреймворками. При

цьому ARKit виступає базовою AR-платформою від Apple, що забезпечує обробку руху, відстеження простору та розпізнавання площин. Для рендерингу сцени ARKit інтегрується з візуалізаційними фреймворками – SceneKit або RealityKit. У цій роботі порівняємо SceneKit і RealityKit та надамо рекомендації щодо їх використання для різних завдань

2.3 Оцінка актуальності та новизни

Проведений огляд та аналіз літературних джерел підтверджують високу актуальність дослідження в сфері використання фреймворків ARKit, SceneKit та RealityKit для розробки AR-додатків на платформі iOS. Сучасні додатки з доповненою реальністю вимагають ефективного управління апаратними ресурсами для забезпечення високої продуктивності, стабільності та реалістичності візуалізації. Зростаючі вимоги до інтерактивності, точності та якості візуалізації роблять питання вибору відповідного фреймворку критично важливим для створення конкурентоспроможних та ефективних AR-додатків.

Аналіз літератури виявив, що кожен із фреймворків має свої сильні та слабкі сторони. ARKit є основою для розробки додатків з доповненою реальністю, забезпечуючи базову функціональність, таку як відстеження руху, виявлення площин та інтеграція з іншими інструментами. SceneKit спеціалізується на рендерингу тривимірної графіки, пропонуючи простий API для створення 3D-сцен, тоді як RealityKit орієнтований на фотореалістичну візуалізацію, підтримку фізичних симуляцій і інтеграцію з просторовим звуком. Однак їх продуктивність залежить від особливостей реалізації та обраного сценарію використання, що стало основою для нашого дослідження.

Новизна роботи полягає у порівнянні за допомогою MCDA продуктивності зв’язки ARKit + SceneKit проти RealityKit для типових сценаріїв AR-додатків. Це дозволяє визначити, який із підходів є оптимальним для конкретних завдань, таких як інтеграція віртуальних об’єктів у реальний світ, обробка складних анімацій або забезпечення фотореалістичності. Використання реальних сценаріїв дозволяє

отримати точні дані щодо FPS, використання CPU та GPU, а також оцінити стабільність і якість візуалізації в динамічних умовах.

Дослідження зосереджене на визначенні як якісних, так і кількісних критеріїв, що дають змогу оцінити фреймворки за ключовими аспектами продуктивності. Особливу увагу приділено стабільності роботи додатків та якості відображення під час виконання різноманітних сценаріїв, які моделюють різні рівні навантаження. Окрім цього, аналіз включає оцінку функціональних можливостей фреймворків, що дозволяє виявити їх сильні та слабкі сторони. Такий підхід сприяє глибшому розумінню потенціалу кожного інструменту та визначенню їх відповідності сучасним вимогам до AR-застосунків.

2.4 Висновки з огляду

Проведена оцінка літературних джерел підтверджує, що дане дослідження є актуальним у сфері порівняння продуктивності фреймворків ARKit, SceneKit та RealityKit. Аналіз зв’язки ARKit + SceneKit та RealityKit відкриває нові можливості для оцінки їхньої ефективності у створенні AR-додатків, забезпечуючи точність вимірювань, адаптивність і здатність ефективно працювати в умовах різних сценаріїв використання. Розробка та впровадження тестових сценаріїв дозволяє суттєво підвищити якість рендерингу, стабільність FPS і оптимізувати використання ресурсів CPU та GPU, що є ключовими аспектами для сучасних AR-додатків.

Дослідження також демонструє високий практичний потенціал запропонованих підходів через їхнє застосування в реальних умовах розробки та тестування AR-додатків. Це підтверджується прикладами успішного використання фреймворків у проєктах для мобільних пристроїв Apple, таких як ігри, освітні додатки та інтерактивні шоуруми. Інтеграція з іншими технологіями, зокрема API Metal, та можливість налаштування функціональності під конкретні потреби проєктів додають цінності нашому дослідженню, роблячи його комплексним та практично значущим. Водночас виявлені прогалини у використанні деяких фреймворків, таких як SceneKit, підкреслюють необхідність подальшого вивчення

їхньої продуктивності та можливостей. Наше дослідження спрямоване на подолання цих недоліків, пропонуючи нові підходи до тестування та оптимізації.

Таким чином, робота створює міцний фундамент для розвитку передових методів оцінки фреймворків ARKit, SceneKit і RealityKit, відповідаючи сучасним вимогам до AR-додатків. Вона відкриває нові напрями для подальших наукових досліджень у сфері оптимізації ресурсів та підвищення продуктивності доповненої реальності на платформі iOS.

3 ПОСТАНОВКА ЗАДАЧІ

Аналіз літературних джерел та проведене дослідження предметної галузі вказують на важливість глибокого розуміння особливостей фреймворків ARKit, SceneKit і RealityKit, які є ключовими інструментами для розробки AR-застосунків на платформі iOS. Ця робота націлена на порівняння фреймворків для створення доповненої реальності під iOS з метою надання рекомендацій розробники щодо вибору конкретного фремворку залежно від потреб проєкту.

Для досягнення зазначеної мети поставлені наступні завдання дослідження:

– провести огляд літературних джерел за обраною тематикою для виявлення існуючих порівнянь фреймворків і підходів до їх аналізу;
– визначити критерії порівняння фреймворків за продуктивністю та функціональними можливостями, а також обмеження їх використання. Для кожного критерію визначити його тип (кількісний чи якісний) та шкалу вимірювання;
– спроектувати та розробити програмний застосунок, що дозволяє реалізувати експерименти для порівняння фреймворків за визначеними критеріями;
– провести експеримент та проаналізувати результати.

Для глибшого розуміння варто розглянути обмеження для фреймворків та сцен. Кожен із фреймворків має специфічні обмеження, які впливають на реалізацію функцій у доповненій реальності. Наприклад, SceneKit обмежений у підтримці сучасних технологій доповненої реальності, тоді як RealityKit краще підходить для створення складних фізичних симуляцій і фотореалістичних сцен. ARKit, у свою чергу, забезпечує високий рівень інтеграції з камерами та трекінгом, але для роботи з 3D-графікою вимагає додаткових інструментів.

Сцена в контексті AR-додатків визначається як тривимірне середовище, у якому розміщуються об’єкти, що взаємодіють між собою та з користувачем. Її параметри включають розмір, текстури, освітлення, анімацію, фізичні ефекти тощо. Від вибору параметрів сцени залежить не лише естетика, але й

продуктивність застосунку, оскільки складні сцени з великою кількістю об’єктів можуть суттєво навантажувати систему.

Тестування фреймворків для розробки AR-додатків вимагає ретельного підходу до створення сценаріїв, які охоплюють різноманітні умови використання. Для забезпечення об’єктивності результатів необхідно враховувати кілька ключових аспектів. По-перше, слід протестувати роботу фреймворків за різних рівнів освітлення: від яскравого денного до слабкого штучного освітлення, оскільки це впливає на точність трекінгу та візуалізацію. По-друге, необхідно оцінити ефективність фреймворків при роботі зі сценами різної складності, наприклад, зі сценами з невеликою кількістю простих об’єктів та з великою кількістю деталізованих моделей.

Крім того, слід враховувати обмеження продуктивності пристроїв. Наприклад, тестування повинно включати як сучасні моделі iOS-пристроїв, так і старіші, щоб визначити стабільність роботи фреймворків на різних апаратних платформах. Інтерактивність також є важливим фактором: сценарії мають враховувати взаємодію користувача з елементами сцени, такі як переміщення об’єктів, зміна їхніх параметрів або додавання нових елементів у режимі реального часу.

Також важливо розглянути різні типи сцен: статичні, з обмеженим числом змін, та динамічні, з активною анімацією чи фізичними ефектами. Це дозволить оцінити не лише базову продуктивність фреймворків, але й їхню здатність адаптуватися до складних умов. Такий підхід допоможе створити повноцінну картину можливостей фреймворків для їх оптимального вибору під конкретні потреби.

Результати дослідження будуть використані для розробки практичних рекомендацій щодо вибору фреймворку залежно від потреб проекту. Вони стануть основою для побудови модульних рішень, які дозволять ефективно адаптувати фреймворки до різних сценаріїв використання, забезпечуючи високу продуктивність і функціональність застосунків.

Таким чином, ця робота спрямована на полегшення та спрощення процесу вибору інструментів для розробки AR-застосунків.

Отримані результати стануть основою для подальших досліджень у галузі доповненої реальності, зокрема щодо оптимізації апаратних ресурсів, підвищення стабільності роботи та розширення функціональних можливостей фреймворків.

4 ТЕОРЕТИЧНЕ ДОСЛІДЖЕННЯ

4.1 Опис інструментів

Даний розділ присвячений детальному аналізу фреймворків, які використовуються для розробки та тестування AR-додатків. У цьому розділі буде розглянуто особливості ARKit, SceneKit і RealityKit, їх функціональність, переваги та обмеження у контексті реалізації проекту. Основна увага приділяється можливостям кожного фреймворку, їх інтеграції один з одним, а також використанню для досягнення високої продуктивності та оптимізації додатків. Особливий акцент зроблено на порівнянні інструментів, їх практичному застосуванні та релевантності до завдань дослідження.

4.1.1 ARKit

ARKit [10] – це фреймворк, розроблений компанією Apple, який дозволяє додаткам на iOS створювати доповнену реальність (AR) шляхом інтеграції цифрового контенту з реальним світом. Основна функція ARKit – це виявлення поверхонь, відстеження положення пристрою у просторі та визначення глибини для розміщення віртуальних об'єктів у реальному середовищі. ARKit може працювати з LiDAR-сканерами, доступними на нових моделях пристроїв iOS. Це дозволяє використовувати тривимірні дані про простір для створення карт глибини в реальному часі.

ARKit має декілька ключових можливостей, серед яких:

– виявлення площин: ARKit може виявляти горизонтальні і вертикальні площини, такі як підлога або стіни, що дозволяє точно позиціонувати віртуальні об'єкти на цих площинах;
– робота з якорями: ARKit дозволяє розпізнавати й обробляти якорі (ARAnchor), які виступають основою для розміщення цифрових об’єктів у реальному просторі. Якорі представляють собою об'єкти, що визначають положення та орієнтацію виявлених елементів відносно фізичного середовища;

– візуально-інерціальна одометрія (Visual Inertial Odometry): ця технологія дозволяє точніше відстежувати рух пристрою, комбінуючи дані з камери та сенсорів (акселерометра, гіроскопа);
– ідентифікування рис обличчя та відстеження їх змін: ARKit може ідентифікувати риси обличчя користувача в реальному часі та надає можливості відстеження їх змін, що відкриває можливості для створення додатків із фільтрами, анімованими аватарами та іншими інтерактивними елементами;
– застосування LiDAR: ARKit обробляє інформацію з LiDAR-сканера для побудови точних моделей поверхонь і об’єктів. Це дає змогу не лише визначати відстань до об’єктів, але й покращувати розміщення цифрових елементів у просторі, забезпечуючи коректне накладання на фізичне середовище;
– ARWorldMap: це функція, яка дозволяє зберігати і відновлювати картографічні дані AR-сцен, що важливо для додатків, де необхідно зберігати стан AR-сесії між різними запусків програми.

ARKit має наступні переваги:

– великий набір API: Apple постійно оновлює ARKit, додаючи нові можливості та покращуючи стабільність роботи;
– широка підтримка на iOS-пристроях: ARKit працює на сучасних iOS-пристроях, починаючи з iPhone 6s, що робить його доступним для широкої аудиторії користувачів;
– інтеграція з іншими фреймворками: ARKit легко інтегрується з SceneKit та RealityKit, дозволяючи використовувати сильні сторони кожного інструменту залежно від завдань проекту.

Обмеження ARKit:

– відсутність можливості підтримки 3D-графіки для моделювання сцен без застосування інших фреймворків;

– високі вимоги до обчислювальних ресурсів: ARKit інтенсивно використовує камеру та процесор, що може негативно впливати на продуктивність на старих пристроях [10].

4.1.2 SceneKit

SceneKit – це потужний фреймворк для 3D-графіки, який пропонує гнучкі можливості для створення та рендерингу складних тривимірних сцен. SceneKit дозволяє розробникам легко створювати 3D-об'єкти, використовувати матеріали, світло, камери, анімації та фізичні симуляції [11]. Хоча SceneKit не спеціалізується на AR, він є одним із основних рендерингових рушіїв для додатків, що використовують ARKit для роботи з тривимірним контентом.

З переваг SceneKit можна виділити:

– гнучкість у роботі з 3D-об'єктами: SceneKit надає широкий інструментарій для створення та маніпуляції 3D-об'єктами, включаючи підтримку складних матеріалів, світла та тіней;
– фізичні симуляції: SceneKit підтримує фізичні симуляції, що дозволяє створювати реалістичні моделі руху та взаємодії об'єктів;
– підтримка анімацій: SceneKit пропонує зручний API для створення та керування анімаціями 3D-об'єктів.

Але SceneKit має значні обмеження порівняно з ARKit та RealityKit, такі як відсутність повної інтеграції з AR: SceneKit самостійно не забезпечує підтримки доповненої реальності, тому для інтеграції з AR потрібно використовувати ARKit. Хоча SceneKit активно використовують для роботи з контентом, 3D-об'єктами, у розробці ігор, високі вимоги до ресурсів: великі та складні 3D-сцени можуть перевантажувати мобільні пристрої, що може негативно позначитися на продуктивності [11].

4.1.3 RealityKit

RealityKit – це сучасний фреймворк, представлений Apple у 2019 році, який був створений для розробки доповненої реальності нового покоління. RealityKit

забезпечує не тільки більш реалістичний рендеринг 3D-об'єктів у реальному світі, але й підтримує фізично коректну взаємодію віртуальних об'єктів з навколишнім середовищем. RealityKit також підтримує LiDAR-сканери на нових пристроях, що дозволяє створювати ще більш точні сцени.

RealityKit має наступні переваги:

– фотореалістичний рендеринг: RealityKit забезпечує високоякісне відображення 3D-об'єктів з реалістичним освітленням, тінями та відображеннями;
– проста робота з фізикою: фреймворк надає простий інтерфейс для створення фізично коректних симуляцій та взаємодій між об'єктами;
– підтримка LiDAR: RealityKit використовує дані LiDAR для точнішого відстеження глибини і розміщення об'єктів, що значно покращує точність та реалістичність доповненої реальності.

Серед обмежень RealityKit можна виділити те, що він вимагає більше ресурсів. У порівнянні з ARKit, RealityKit є більш вимогливим до апаратних ресурсів, що може негативно впливати на продуктивність на старих пристроях [12].

4.1.4 Сумісне використання ARKit та SceneKit

ARKit та SceneKit окремо пропонують потужні можливості, проте їхні функціональні обмеження стають очевидними при створенні комплексних AR-додатків. ARKit забезпечує точне виявлення поверхонь, трекінг рухів і створення якорів для позиціювання об’єктів у фізичному середовищі. Водночас SceneKit дозволяє працювати з 3D-графікою, створювати об’єкти, текстури та анімації, проте самостійно не взаємодіє з реальним середовищем.

ARKit не має вбудованих засобів для створення та рендерингу 3D-об’єктів, а SceneKit не може сканувати навколишнє середовище чи забезпечувати інтеграцію в реальний простір. Тому автор пропонує надалі розглядати ARKit та SceneKit як зв’язку. Разом ці інструменти компенсують обмеження один одного. Завдяки використанню зв’язки розробник отримує переваги обох фреймворків.

Серед недоліків зв’язки ARKit та SceneKit можна виділити:

– обмежена підтримка складних 3D-сцен: Рендеринг складних сцен у реальному часі може створювати значне навантаження на систему;
– обмеження SceneKit: Незважаючи на можливість створення 3D-графіки, SceneKit поступається RealityKit у фотореалістичності та фізичному моделюванні;
– Потреба в інтеграції: Розробник повинен самостійно налаштовувати взаємодію між SceneKit та ARKit, що вимагає додаткового часу та зусиль.

4.2 Загальновідомі критерії оцінювання фреймворків

Дослідження використання інструментів передбачає спостереження за продуктивністю виконання тих чи інших операцій з 3D-об’єктами всередині сцени з застосуванням зв’язки ARKit + SceneKit або самостійно RealityKit. Для аналізу їхньої продуктивності необхідно створити AR-додаток, який використовує кожен із цих фреймворків для роботи зі 3D-сценами. Спостереження проводиться за такими аспектами як взаємодія з 3D-об'єктами, завантаження сцен, функціональність анімацій. Обов’язково буде проаналізовано, як різні фреймворки справляються з анімацією об'єктів та їх фізичними взаємодіями, бо це основний функціонал майже кожного AR-додатку. Основну увагу буде приділено стабільності роботи додатків при великій кількості об'єктів та складних анімаціях.

Продуктивність AR-додатків вимірюється за кількома основними критеріями. Першочергово це FPS (Frames Per Second) – частота кадрів в секунду. Вона є одним із найважливіших показників продуктивності, бо дозволяє оцінити плавність роботи додатка. Високе значення FPS забезпечує плавний користувацький досвід, а низьке значення може призвести до ривків в анімації та погіршення загального враження від роботи додатку. Оптимальний рівень для VR-додатків – це стабільні 60 FPS. Вимірювання FPS дає змогу оцінити, наскільки фреймворки справляються з візуалізацією сцени та підтримують високу продуктивність під час анімацій [13].

Критичним для загальної продуктивності програми є використання CPU (центрального процесора) під час виконання VR/AR функцій. CPU в таких додатках

відповідає за обробку даних і виконання інструкцій, пов'язаних із тривимірною графікою, відстеженням рухів, фізикою об'єктів і керуванням анімаціями. CPU бере участь у створенні та управлінні сценою. Це включає завантаження 3D-об'єктів, застосування текстур і матеріалів, а також розміщення об'єктів у віртуальному просторі. Хоча частина роботи делегується GPU, CPU відповідає за підготовку і передачу інструкцій для відображення об'єктів. Також CPU обробляє всі фізичні взаємодії віртуальних об'єктів, як-от зіткнення, гравітація та рух. Особливо важливо це в ігрових VR-додатках, де реалістична поведінка об'єктів підвищує якість взаємодії. У VR-додатках використовується багато датчиків, як-от акселерометр, гіроскоп і камера, для відстеження руху користувача та його положення в просторі. CPU відповідальний за інтеграцію цих даних та їх подальше використання для оновлення сцени в режимі реального часу. Важливо розуміти, що саме CPU керує запуском і зупинкою анімацій для об'єктів або інші трансформації в тривимірному просторі. При цьому важливо підтримувати баланс між плавністю анімації та мінімальним використанням ресурсів [14].

Використання GPU (графічного процесора) в AR/VR-додатках є ключовим аспектом для досягнення високої якості графіки та плавності взаємодії в реальному часі. GPU спеціалізується на виконанні обчислень, пов'язаних із рендерингом зображень, тому саме цей компонент виконує більшість завдань, пов'язаних із відтворенням тривимірних сцен і анімацій. Головна роль GPU полягає у рендерингу графічних елементів, тобто відтворенні 3D-об'єктів у сцені на екрані. Чим складніша сцена, тим більше обчислювальної потужності потрібно від GPU. У AR-додатках GPU допомагає відображати віртуальні об'єкти у реальному просторі, поєднуючи їх з зображенням, отриманим через камеру. GPU відповідає за обробку текстур, накладення тіней, відображення освітлення, рефлексій і рефракцій. Для реалістичності AR-сцен ці елементи критично важливі, оскільки саме вони визначають, наскільки гармонійно віртуальні об'єкти будуть виглядати в реальному середовищі. GPU також може допомагати з фізичними симуляціями об'єктів, що включають обчислення взаємодії об'єктів у віртуальному світі (наприклад, гравітація, зіткнення об'єктів). Це робить взаємодію з AR-об'єктами більш

природною та реалістичною. Хоча GPU ефективно справляється з графічними обчисленнями, перевантаження може призвести до зниження FPS (кадрів на секунду) і загального погіршення користувацького досвіду. Наприклад, при надто великій кількості об'єктів або анімацій GPU може не встигати відтворювати їх на належному рівні. Для VR/AR-додатків важливо, щоб зображення відтворювалося з високою частотою кадрів (60 FPS або більше). Якщо GPU не справляється із завданням, це може призвести до «затримок» в інтерфейсі або анімаціях, що знижує якість взаємодії з доповненою реальністю. Надмірне використання GPU може спричинити швидке розрядження батареї або перегрів пристрою. Тому важливо забезпечити збалансовану роботу GPU, щоб підтримувати продуктивність, але водночас не споживати зайві ресурси [15]. Адаптивність інструментів дає можливість розробникам враховувати обмеження пристроїв і зменшувати навантаження на ресурси [16], зокрема GPU та CPU.

4.3 Визначення та аналіз додаткових критеріїв порівняння фреймворків, які впливають на користувацький досвід

Розробка критеріїв для оцінки фреймворків ARKit, SceneKit і RealityKit є важливим кроком у порівнянні їхніх можливостей для вибору оптимального інструменту залежно від потреб проекту. У цьому розділі представлено додаткові критерії, за якими буде проводитися порівняння, а також пояснюється доцільність обраних критеріїв, як якісних, так і кількісних. Для кожного з якісних критеріїв треба визначити шкалу його вимірювання.

Одним із головних критеріїв є функціональність, яка визначає, наскільки фреймворк забезпечує базові та розширені можливості для розробки AR/VR-сцен. Сюди входить підтримка популярних форматів 3D-моделей (GLTF, OBJ, FBX), наявність інструментів для обробки фізичних симуляцій, рівень масштабованості складних сцен, а також легкість кастомізації і роботи з освітленням, текстурами та анімаціями. Ці параметри здебільшого оцінюються якісно, оскільки наявність певних функцій легко визначити з документації фреймворка, а їх якість – на основі експериментального тестування.

Простота використання є важливим критерієм, який впливає на швидкість і зручність розробки AR-додатків. Для оцінки цього аспекту враховуються кілька ключових параметрів, таких як доступність документації, інтуїтивність API, підтримка інтеграції з іншими фреймворками та наявність інструментів для налагодження.

Першим параметром є доступність документації, яка має бути чіткою, детальною та зрозумілою. Документація повинна містити приклади використання ключових функцій, опис можливих помилок і способів їх вирішення. Другим параметром є інтуїтивність API, яка визначає, наскільки легко розробникам використовувати інструменти фреймворку без необхідності тривалого навчання. Інтуїтивне API має мінімізувати кількість викликів і забезпечувати логічну структуру команд та методів.

Також враховується підтримка інтеграції з іншими фреймворками та інструментами. Це дозволяє розробникам легко комбінувати різні технології для створення комплексних додатків. Окрім цього, важливою є наявність інструментів для налагодження, які дозволяють швидко ідентифікувати та виправляти помилки. Інструменти мають бути інтегровані у середовище розробки та забезпечувати зручний доступ до логів і діагностичних даних.

Обмеження фреймворків визначають ті аспекти, які прямо впливають на практичне використання бібліотек у проєктах з доповненою реальністю. Це, насамперед, сумісність із різними версіями операційних систем і здатність ефективно працювати зі складними сценами, які містять велику кількість об’єктів або мають високу деталізацію.

Сумісність фреймворків визначає, на яких пристроях та версіях операційної системи можна запускати додаток. Це важливо враховувати, оскільки від сумісності залежить потенційна аудиторія та комерційний успіх продукту.

Масштабованість складних сцен означає здатність фреймворку підтримувати стабільну роботу за великої кількості об’єктів, динамічного освітлення та високої деталізації моделей.

Ці обмеження відіграють важливу роль у виборі фреймворку під конкретний проєкт. Вони допомагають розробникам краще планувати архітектуру застосунку, уникати технічних бар’єрів на різних пристроях і вибирати оптимальний шлях досягнення балансу між продуктивністю та якістю графіки.

Анімації є ключовим елементом для створення динамічних і живих сцен у додатках доповненої реальності. Вони дозволяють наочно відображати процеси руху, зміни станів та взаємодії об'єктів, що важливо для досягнення реалістичності та інтуїтивного сприйняття користувачем. Анімації можуть підвищувати привабливість інтерфейсу, а також допомагають користувачу краще орієнтуватися у віртуальному середовищі, створюючи відчуття природного руху та взаємодії. Відсутність або обмеженість інструментів для анімації значно знижує загальну якість графіки та обмежує функціонал AR-додатку.

Фотореалізм, у свою чергу, відображає здатність фреймворку створювати зображення, максимально наближені до реальності. Це включає підтримку фізично коректного рендерингу (PBR), динамічного освітлення, якісних матеріалів та складних шейдерів. Високий рівень фотореалізму дозволяє досягти ефекту «занурення» користувача у змішане середовище, де віртуальні об'єкти виглядають органічно та не відрізняються від реальних. Це важливо як для естетики та сприйняття додатку, так і для функціональності – наприклад, у додатках для проєктування, де важливо візуалізувати кінцевий результат у реальному середовищі.

Фізичні симуляції забезпечують правдоподібність взаємодії об'єктів, дозволяючи точно відтворювати сили, рухи, зіткнення та інші ефекти, характерні для реального світу. Завдяки підтримці фізичних рушіїв користувач отримує більш достовірний досвід: об'єкти реагують на дотики, прискорення, сили тяжіння чи взаємодію один з одним. Це особливо важливо у навчальних або інженерних застосунках, де необхідно моделювати реальні процеси або перевіряти поведінку об'єктів у конкретних умовах. Висока якість фізичних симуляцій підвищує достовірність сцени та додає додатку нові рівні функціональності й цінності для користувача.

Оптимізація продуктивності передбачає засоби для управління складністю сцени, зменшення кількості обчислень та підвищення швидкодії. Це особливо важливо для AR-застосунків, які повинні працювати в реальному часі та забезпечувати плавність анімацій та інтерактивності. SceneKit, наприклад, підтримує LOD (Level of Detail) – підхід, що дозволяє автоматично знижувати деталізацію віддалених об’єктів, а також інструменти для профілювання продуктивності. RealityKit, натомість, використовує ECS-підхід (Entity-Component-System), що полегшує організацію об’єктів та їхню оптимізацію, проте має менше інструментів для глибокої ручної оптимізації. Наявність чи відсутність цих функцій визначає, наскільки додаток буде стабільним під великим навантаженням і як легко розробник зможе підтримувати необхідний рівень FPS.

Кастомізація – це можливість налаштувати вигляд і поведінку об’єктів у сцені під власні потреби. Вона включає такі аспекти, як підтримка кастомних шейдерів, можливість створення унікальних матеріалів, управління освітленням та використання фізично коректного рендерингу (PBR). SceneKit має розвинуті можливості для кастомізації: підтримку PBR, гнучке налаштування матеріалів та шейдерів, що дозволяє досягти високого рівня деталізації та реалістичності сцени. RealityKit, у свою чергу, надає базовий набір інструментів, орієнтований на швидкий старт та простоту, але з меншими можливостями для детальної кастомізації. Вибір між цими підходами впливає на те, наскільки індивідуальним та адаптованим під задачу буде фінальний продукт.

Загалом, важливість цих критеріїв у розробці AR-додатків полягає в тому, що вони допомагають знаходити баланс між продуктивністю та гнучкістю. Ретельне налаштування продуктивності дозволяє досягти стабільної роботи навіть у складних сценах, тоді як кастомізація відкриває шлях до унікального користувацького досвіду, що є особливо цінним у проєктах з високими вимогами до якості графіки та індивідуального стилю.

Час взаємодії є важливим показником продуктивності фреймворків, який визначає швидкість виконання основних операцій під час роботи з AR-додатками. Цей критерій дозволяє оцінити, наскільки швидко система реагує на дії розробника

або користувача, а також визначити затримки при виконанні ключових завдань, таких як ініціалізація сцени, імпорт моделей або запуск анімацій. Наприклад, час завантаження сцени дозволяє оцінити, як ефективно фреймворк працює з великими та складними сценами, включаючи текстури, освітлення та об’єкти. Час імпорту 3D-моделей демонструє здатність фреймворку обробляти популярні формати, такі як USDZ, GLTF або FBX, і впливає на загальну швидкість розробки.

Ще одним ключовим показником є час виконання анімацій, який визначає, наскільки плавно та без затримок анімації відображаються на екрані. У випадку інтерактивних додатків для доповненої реальності цей показник є критично важливим, адже користувацький досвід значною мірою залежить від швидкості відгуку системи.

Оцінка часу взаємодії дозволяє порівняти ефективність фреймворків у виконанні базових операцій, а також виявити їхні слабкі сторони. Наприклад, фреймворк із високою затримкою завантаження може бути менш ефективним у додатках, що вимагають швидкої ініціалізації сцен. Такий аналіз допомагає визначити, який фреймворк краще підходить для конкретних завдань, і забезпечує обґрунтованість вибору технології для розробки AR-додатків.

Для зручності порівняння критеріїв можна створити таблицю, в якій кожен параметр буде розбитий за типом оцінки (кількісний чи якісний) (див. табл. 4.1).

Таблиця 4.1 – Порівняння критеріїв (таблиця виконана самостійно)

Критерій	Тип оцінки	Примітка
Підтримка форматів 3D-моделей	Наявність по кожному з форматів	GLTF, OBJ, FBX, USDZ, USD
Інтеграція з AR	Наявність по кожній з операцій	Жести, трекінг рухів, фізичні симуляції, тощо
Продуктивність	Кількісний	Кількість кадрів за секунду (FPS), навантаження на CPU/GPU

Кінець таблиці 4.1

Критерій	Тип оцінки	Примітка
Налаштування та кастомізація	Ступінь	Наскільки гнучко можна змінювати матеріали, анімації, налаштовувати графіку
Обмеження	Ступінь	Сумісність із платформами, можливість масштабування складних сцен
Простота використання	Ступінь	Дружній API, документація, тощо

Тип оцінки “ступінь” є якісним і буде формалізований для кожного критерія у наступних розділах.

4.4 Формалізація критеріїв

Для якісних критеріїв введемо шкалу для ранжування фреймворків по зазначених критеріях. Зокрема, розглянемо критерії, які мають тип оцінки ступінь.

4.4.1 Графічні можливості фреймворків

Графічні можливості фреймворків у даній роботі розглядаються як сукупність інструментів і технологій, що дозволяють створювати візуально привабливі та реалістичні сцени у додатках доповненої реальності. До цієї групи критеріїв входять анімації, які забезпечують плавність руху та реалістичність взаємодії об’єктів; фізичні симуляції, що дозволяють відтворювати закони фізики у тривимірному просторі, створюючи ефект природної взаємодії; а також фотореалізм, який відображає здатність фреймворків досягати максимальної відповідності між віртуальними об’єктами та реальним світом. Усі ці параметри мають важливе значення для створення переконливого користувацького досвіду та

визначають, наскільки ефективно фреймворк дозволяє реалізувати задуманий візуальний стиль і якість графіки у проєкті.

Анімація є одним із ключових аспектів візуального досвіду у розробці AR-додатків, оскільки саме вона надає сценам динамічності, живості та інтерактивності. Від рівня підтримки анімацій залежить не тільки естетичне враження користувача, а й зручність подачі інформації, плавність переходів між станами об'єктів, а також можливість інтеграції з іншими ефектами або поведінкою сцени. Введемо наступну шкалу:

- 0 – відсутні або мінімальні можливості для роботи з анімацією;
- 1-2 – є лише базові анімаційні переходи без налаштування;
- 3-4 – стандартні засоби керування анімаціями, з базовим контролем швидкості та повторів, але без гнучких сценаріїв;
- 5-6 – доступна робота з ключовими кадрами, покадровими анімаціями та базовою кастомізацією (зміна кривих інтерполяції, контроль циклів);
- 7 – сучасний інструментарій для динамічних ефектів та інтеграції з AR-середовищем, зручне управління стандартними анімаціями, але обмежена гнучкість для складних кастомних сценаріїв;
- 8 – повноцінні можливості для складних покадрових і процедурних анімацій, гнучке налаштування та інтеграція зі сторонніми редакторами;
- 9 – підтримка професійних механізмів (інверсна кінематика, розширені фізичні анімації);
- 10 – максимальна свобода, масштабна бібліотека інструментів і можливість налаштування будь-яких аспектів анімаційної системи.

Фотореалізм є важливим критерієм для оцінки можливостей фреймворків у створенні візуально привабливих та реалістичних AR-сцен. Цей показник відображає здатність рушія або фреймворку працювати з фізично коректними матеріалами, налаштовувати освітлення та відтворювати складні ефекти, які максимально наближують зображення до реального світу. Він особливо критичний у додатках, де потрібна точна візуалізація, наприклад, для демонстрації продуктів або моделювання складних фізичних процесів.

Введемо наступну шкалу:

– 0-2 – відсутні засоби реалістичного освітлення чи матеріалів; сцена виглядає спрощено;
– 3-4 – мінімальна підтримка базових 3D-ефектів: просте освітлення і текстури без реалістичності;
– 5-6 – є різноманітні типи освітлення і базові тіні, але немає підтримки PBR та складних шейдерів;
– 7-8 – підтримка PBR, динамічного освітлення, базової постобробки (ефекти, глибина різкості); вже можливо досягати переконливого реалізму;
– 9-10 – розвинуті фотореалістичні інструменти: продумана робота з PBR, складне динамічне освітлення, повноцінна постобробка; сучасні методи трасування променів, глобальне освітлення, можливість рендерингу сцен.

Фізичні симуляції є важливою характеристикою фреймворків для AR-застосунків, оскільки дозволяють досягти більш реалістичної взаємодії об’єктів у сцені, моделювати поведінку тіл у просторі та враховувати фізичні властивості. Така підтримка значно підвищує достовірність візуалізації та забезпечує занурення користувача у віртуальне середовище. У цьому критерії оцінюється, наскільки фреймворк дозволяє працювати з жорсткими та м’якими тілами, налаштовувати матеріали, взаємодії, обмеження (constraints) та розширені фізичні ефекти.

Введемо наступну шкалу:

– 0-2 – відсутність або мінімальна підтримка фізичних взаємодій, обмежені або спрощені колізії;
– 3-4 – початкові можливості: прості налаштування маси, тертя, базові констрейнти (hinge, fixed). Обмежена робота з матеріалами та силами;
– 5-6 – розвинуті інструменти для жорстких тіл, налаштування матеріалів і простих колізійних форм; базові події зіткнення та деякі констрейнти;
– 7-8 – повноцінна підтримка складних фізичних сценаріїв, констрейнтів і процедурного управління . Можливе поєднання анімацій і фізики;

– 9-10 – максимальні можливості: детальна робота з м’якими тілами, рідинами, передові констрейнти (ragdoll, gear, nested). GPU-акселерація, розширення чи заміна рушія для спеціалізованих фізичних розрахунків.

4.4.2 Налаштування та кастомізація

Налаштування та кастомізація – це один з ключових аспектів роботи з графічними фреймворками, який визначає здатність адаптувати їх під конкретні вимоги проєкту та максимально ефективно використовувати ресурси пристрою. Ці можливості впливають не лише на візуальну якість сцени, але й на швидкість роботи додатка, забезпечуючи баланс між високою продуктивністю та гнучкістю графічного рендерингу. Наявність продуманих інструментів для налаштування та кастомізації дозволяє розробнику врахувати всі деталі проєкту та оптимізувати роботу програми під конкретні умови.

Оптимізація продуктивності відображає, наскільки фреймворк дозволяє ефективно налаштовувати продуктивність проєкту. Йдеться про наявність інструментів для управління рівнями деталізації (LOD), профілювання, оптимізації рендерингу та ресурсів (наприклад, через розподіл об’єктів у сцені чи спрощення ієрархії).

Введемо наступну шкалу:

– 0-2 – відсутність або мінімальні засоби оптимізації (загальні налаштування системи);
– 3-4 – базова оптимізація (лише загальні налаштування LOD чи схожі інструменти);
– 5-6 – середній набір засобів оптимізації (LOD, базове профілювання);
– 7-8 – розширена оптимізація: детальне управління LOD, профілювання через інструменти розробника, гнучке налаштування;
– 9-10 – максимальна гнучкість оптимізації: інструменти для глибокої кастомізації та профілювання, можливість власної інтеграції механізмів керування продуктивністю.

Кастомізація як критерій оцінює, наскільки легко і глибоко можна налаштувати графіку, анімації та інші аспекти застосунку у рамках фреймворку. Це стосується роботи з кастомними шейдерами, матеріалами (PBR), освітленням та іншими елементами сцени.

Введемо наступну шкалу:

– 0-2 – немає можливості або дуже базова кастомізація;
– 3-4 – лише прості зміни кольору/матеріалу без глибокого налаштування;
– 5-6 – середній рівень: підтримка кількох типів матеріалів або базових налаштувань шейдерів;
– 7-8 – розвинуті можливості для створення власних матеріалів, шейдерів, повноцінна робота з PBR та освітленням;
– 9-10 – максимальна гнучкість: розширюваність через плагіни або власні модулі, можливість реалізації складних кастомних ефектів, глибоке управління матеріалами і сценами.

4.4.3 Простота використання

Простота використання є складовим критерієм, який відображає, наскільки зручно розробникам працювати з фреймворками та наскільки ефективно вони можуть реалізувати потрібну функціональність. Цей критерій включає чотири показники: наявність спільноти, якість документації та прикладів, наявність редактору сцени, а також рівень організації сцени та необхідність оптимізації.

Спільнота, документація, приклади оцінюються за повнотою та доступністю інформації для розробників. Введемо наступну шкалу:

– 0-2 – відсутня або дуже бідна документація, немає зразків коду, ком’юніті практично відсутнє;
– 3-4 – є базова документація, мінімальні приклади, ком’юніті невелике;
– 5-6 – документація є, але не надто структурована чи обмежена; ком’юніті середнє, підтримка присутня, але не глибока;
– 7-8 – гарна документація з прикладами та активне ком’юніті, яке допомагає знаходити рішення;

– 9-10 – дуже детальна документація, велика база прикладів, активна спільнота, регулярне оновлення матеріалів.

Редактор сцени відображає можливості редактора (якщо він є) для створення та налаштування сцен. Введемо наступну шкалу:

– 0-2 – відсутній або практично не функціонує;
– 3-4 – базовий, дозволяє виконати мінімальні налаштування, але має серйозні обмеження;
– 5-6 – є, але потребує багато часу для освоєння, функціонал обмежений;
– 7-8 – зручний, дозволяє будувати сцени з різним рівнем складності, підтримує потрібні інструменти;
– 9-10 – потужний редактор з широкими можливостями, що дозволяє швидко створювати та налаштовувати сцени.

Організація сцени показує, наскільки зручно будувати структуру сцени (наприклад, вузлова структура або ECS-підхід). Введемо наступну шкалу:

– 0-2 – дуже незручна, хаотична структура, відсутність стандартних підходів;
– 3-4 – є базова підтримка організації, але мало інструментів для управління складними сценами;
– 5-6 – середній рівень: можна працювати, але не вистачає гнучкості або сучасних підходів;
– 7-8 – зручна та зрозуміла організація, підтримує більшість типових завдань;
– 9-10 – сучасний підхід (наприклад, ECS), чудово підходить для складних ієрархічних структур, забезпечує високу керованість.

Необхідність оптимізації характеризує, наскільки потрібні додаткові ручні налаштування для досягнення високої продуктивності. Введемо наступну шкалу:

– 0-2 – фреймворк сильно «гальмує» без оптимізації, потрібні складні та великі доопрацювання;
– 3-4 – потребує багато ручної роботи для забезпечення стабільної роботи навіть із середніми сценами;

– 5-6 – для важких сцен потрібна ручна оптимізація, але базова продуктивність прийнятна;
– 7-8 – автоматична оптимізація працює добре, ручна робота потрібна тільки для складних випадків;
– 9-10 – повноцінна автоматична оптимізація, мінімум необхідності ручних втручань, висока стабільність продуктивності.

4.4.4 Обмеження фреймворків

Важливо враховувати обмеження фреймворків. Для цього було виділено такі критерії. Перший критерій – сумісність. Цей показник визначає, на яких версіях операційних систем iOS, macOS та visionOS доступний фреймворк. Висока сумісність означає ширші можливості для розробників та кращий охват аудиторії, адже додаток зможе працювати на більшій кількості пристроїв. Введемо таку шкалу:

– 0-2 бали – дуже вузька сумісність: підтримка лише однієї версії або необхідність значних доробок для запуску;
– 3-4 бали – підтримка тільки для дуже обмеженого кола пристроїв (наприклад, тільки для iOS, без macOS чи visionOS);
– 5-6 балів – обмежена підтримка: лише для новіших версій операційних систем або окремих пристроїв;
– 7-8 балів – підтримка більшості версій, мінімальні обмеження (наприклад, тільки для iOS 13+, macOS 10.15+);
– 9-10 балів – підтримка усіх основних версій iOS, macOS та visionOS без обмежень.

Другий критерій – масштабованість складних сцен. Цей показник оцінює, наскільки добре фреймворк здатен працювати зі сценами, які містять велику кількість об'єктів чи складні ієрархії. Висока масштабованість важлива для застосунків, що потребують рендерингу детальних моделей або одночасної взаємодії з багатьма об’єктами. Задамо таку шкалу:

– 0-2 бали – не підходить для складних сцен: значні проблеми з продуктивністю або взагалі не працює;
– 3-4 бали – здатен працювати тільки зі спрощеними сценами, значні обмеження для складних моделей;
– 5-6 балів – справляється з помірно складними сценами, потребує додаткової оптимізації для більшої складності;
– 7-8 балів – ефективно працює зі складними сценами, мінімальні потреби оптимізації;
– 9-10 балів – Висока ефективність роботи навіть зі сценами великої складності без додаткової оптимізації.

4.5 Порівняння фреймворків за якісними критеріями

В рамках дослідження продуктивності SceneKit і RealityKit проведемо теоретичний аналіз їхніх можливостей за визначеними критеріями, які є ключовими для оцінки функціональності, гнучкості, легкості використання та інших аспектів, необхідних для розробки AR-додатків. Оцінювання проводимо за тими параметрами, які можна оцінити без експериментальних замірів, ґрунтуючись на експертних оцінках. Для виставлення оцінок фреймворків за обраними критеріями було залучено 5 експертів (колег по роботі, які мають досвід у розробці AR-додатків). Оцінки були узгоджені з використанням методу Delphi [17].

RealityKit нативно підтримує GLTF, OBJ, USDZ та USD, що дозволяє легко працювати з сучасними форматами 3D-моделей. Водночас підтримка FBX відсутня, оскільки цей формат орієнтований на більш складні анімації, які зазвичай обробляються у професійних інструментах.

SceneKit забезпечує широкий спектр підтримки, включаючи GLTF, OBJ, FBX та USDZ, але не підтримує формат USD без додаткової обробки. Це обмежує можливості роботи з деякими сучасними 3D-ресурсами.

Так як раніше була зазначена шкала від 1 до 10, то відповідно підтримується визначається як 10 балів, не підтримується - 1 бал.

Результати аналізу підтримки форматів можна побачити в таблиці 4.2.

Таблиця 4.2 – Порівняння бібліотек за підтримкою зазначених форматів (таблиця виконана самостійно)

Формат	SceneKit	RealityKit
GLTF	10 (підтримується)	10 (підтримується)
OBJ	10 (підтримується)	10 (підтримується)
FBX	10 (підтримується)	1 (не підтримується)
USDZ	10 (підтримується)	10 (підтримується)
USD	1 (не підтримується)	10 (підтримується)

Проведемо оцінку графічних можливостей фреймворків SceneKit та RealityKit за виділеними раніше критеріями, використовуючи узгоджені шкали. Для кожного з критеріїв надано окрему оцінку, яка враховує як кількісні, так і якісні аспекти роботи фреймворків.

Перший показник – підтримка форматів 3D-файлів. Обидва фреймворки отримали оцінку 8, оскільки вони мають хороший рівень сумісності з основними форматами (GLTF, OBJ, FBX, USDZ, USD). Однак SceneKit не підтримує USD, а RealityKit – FBX, що відображено у відповідному описі.

Наступний критерій – анімації. SceneKit отримав оцінку 8, оскільки він забезпечує достатню підтримку покадрових і процедурних анімацій, хоча не має реальної анімації в реальному часі. RealityKit оцінено на 7 через кращу інтеграцію інверсної кінематики та сучасні механізми керування анімацією, однак складні анімаційні сценарії вимагають додаткових рішень.

Третій показник – фотореалізм. SceneKit отримав оцінку 7, що свідчить про його добрі можливості, але для досягнення фотореалістичної якості потрібні ручні налаштування та більше часу на оптимізацію. RealityKit оцінено на 8 завдяки автоматичному освітленню, зручнішому керуванню матеріалами та інтеграції зі справжнім середовищем через ARKit.

Четвертий критерій – фізичні симуляції. SceneKit отримав оцінку 7, оскільки має базовий фізичний рушій і підтримку кількох видів колізій, що достатньо для більшості застосунків. RealityKit отримав оцінку 6 – хоча він підтримує базову фізику, його можливості обмежені простими тілесними взаємодіями без глибокої кастомізації. Висновки зведено у таблицю 4.3.

Таблиця 4.3 – Оцінка графічних критеріїв для RealityKit та SceneKit (таблиця виконана самостійно)

Показник	SceneKit	RealityKit	Опис
Формати 3D-файлів (GLTF, OBJ, FBX, USDZ, USD)	8	8	SceneKit не підтримує формат USD, RealityKit не підтримує FBX
Анімації	8	7	SceneKit не підтримує анімацію в реальному часі, RealityKit – інверсну кінематику та стани анімацій
Фотореалізм	7	8	SceneKit потребує ручного налаштування, RealityKit має автоматичне освітлення
Фізичні симуляції	7	6	SceneKit має базову фізику, RealityKit – прості тілесні взаємодії

Проведемо оцінку фреймворків SceneKit і RealityKit за критеріями налаштування та кастомізації, яка враховує можливості оптимізації продуктивності та кастомізації графічних компонентів.

SceneKit отримав оцінку 8 по оптимізації продуктивності завдяки підтримці механізмів управління рівнем деталізації (LOD), профілювання та оптимізації складних сцен. Це дозволяє розробникам зменшувати навантаження на пристрій і досягати плавної роботи додатків. RealityKit отримав оцінку 5, оскільки він

обмежується структурою ECS, яка автоматично оптимізує графічні сцени, але не надає настільки широких можливостей контролю та налаштувань, як у SceneKit.

В кастомізації SceneKit отримав оцінку 9 за свою здатність підтримувати шейдери, PBR-матеріали та розширені налаштування освітлення й текстур. Це важливо для створення унікальних графічних ефектів, кастомних матеріалів і забезпечення гнучкості дизайну. RealityKit отримав оцінку 5 через більш базову підтримку кастомізації: він забезпечує лише базові налаштування матеріалів і освітлення без можливості глибокої кастомізації.

Оцінимо обмеження фреймворків SceneKit і RealityKit, що допомаже визначити їхню сумісність з різними версіями ОС та здатність до масштабування складних сцен.

За сумістність SceneKit отримав оцінку 10 завдяки повній підтримці всіх версій iOS, macOS та visionOS, що робить його універсальним інструментом для розробки AR-додатків на різних пристроях Apple. RealityKit отримав оцінку 9 через те, що він доступний тільки для iOS 13+, macOS 10.15+ та visionOS, що обмежує його використання на старіших пристроях.

Розглянемо масштабованість складних сцен. SceneKit отримав оцінку 7, оскільки хоча він добре справляється з більшістю сцен, для складних сценаріїв потрібне додаткове профілювання та оптимізація. RealityKit отримав оцінку 9, адже він забезпечує ефективну роботу навіть зі складними сценами завдяки використанню Metal API, що дозволяє зберігати високу продуктивність і плавність взаємодії.

Проаналізуємо легкість використання зазначених комбінацій бібліотек.

ARKit відзначається простим у використанні API з великою кількістю навчальних матеріалів. Це робить його доступним для розробників, які тільки починають працювати з AR-додатками. SceneKit має більш складний API, але завдяки широкій документації та інтеграції з інструментами розробки iOS він залишається зрозумілим для більшості розробників. SceneKit має певні складнощі через необхідність ручної інтеграції між фреймворками. Документація для кожного фреймворку окремо є якісною, але відсутність цілісного керівництва для роботи зі

зв’язкою створює додаткові труднощі для розробників. Спільнота активна, матеріалів, що охоплюють саме інтеграцію ARKit та SceneKit, достатньо. Тож легкість використання для зв’язки можна оцінити добре.

RealityKit пропонує інтуїтивний API, оптимізований для роботи з AR-додатками. Велика кількість готових інструментів і шаблонів дозволяє швидко створювати складні сцени. Його основною перевагою є простота використання при розробці фотореалістичних AR-додатків. Легкість використання фреймворку оцінюється високо. Інтуїтивне API та детальна документація значно полегшують роботу розробників. Активна спільнота та велика кількість туторіалів сприяють швидкому вирішенню проблем і впровадженню нових функцій.

Порівняння фреймворків можна побачити в таблиці 4.4.

Таблиця 4.4 – Порівняння бібліотек за визначеними критеріями (таблиця виконана самостійно)

Критерій	SceneKit	RealityKit	Опис
Оптимізація продуктивності	6	5	SceneKit підтримує LOD та профілювання, RealityKit – ECS підхід
Кастомізація	9	5	SceneKit підтримує шейдери та PBR, RealityKit – базове налаштування
Сумісність	10	9	SceneKit – для всіх версій iOS/macOS/visionOS; RealityKit – з iOS 13+ / macOS 10.15+
Масштабованість складних сцен	7	9	SceneKit потребує оптимізації; RealityKit – ефективний завдяки Metal

Кінець таблиці 4.4

Критерій	ARKit + SceneKit	RealityKit	Опис
Спільнота, документація, приклади	9	8	Обидва фреймворки мають гарну документацію
Структура та побудова сцени в коді	8	8	В обох фреймворках сцена створювалась вручну через код
Необхідність оптимізації	5	9	SceneKit потребує ручного налаштування, RealityKit – автоматична оптимізація

Таким чином, кожен із фреймворків має свої сильні сторони, які залежать від специфіки використання. ARKit + SceneKit забезпечує базову функціональність для розробки AR-додатків, але потребує більше зусиль для реалізації складних сцен. Обмеження в освітленні та фізиці роблять зв’язку менш гнучкою для фотореалістичних або інтенсивно інтерактивних проектів. RealityKit демонструє вищу гнучкість завдяки вбудованій підтримці сучасних форматів, реалістичного освітлення, фізичних симуляцій та складних анімацій.

4.6 Виділення показників, що впливають на оцінку продуктивності фреймворків за різних сценаріїв використання

Для об’єктивної оцінки фреймворків необхідно виділити низку параметрів, що впливають на їхню ефективність у різних сценаріях використання. Ці параметри дозволяють визначити, наскільки фреймворк відповідає вимогам проєкту та які аспекти потребують оптимізації. На основі цих параметрів будуть ґрунтуватися сценарії тестування.

Першим важливим параметром є складність сцени, яка включає кількість об’єктів, полігональність моделей і текстури. Більш складні сцени з високодеталізованими моделями та складними матеріалами вимагають значних

ресурсів, тому фреймворки повинні забезпечувати стабільну продуктивність навіть за таких умов. Наприклад, сцена з 50 динамічними об’єктами, що мають текстури в роздільній здатності 4K, є серйозним викликом для фреймворку.

Другим параметром є анімації. Одночасне відтворення складних анімацій, зокрема руху об’єктів, фізичних симуляцій і взаємодії, також суттєво впливає на продуктивність. Фреймворк має підтримувати плавність анімацій навіть при інтенсивному навантаженні сцени.

Одним із ключових параметрів, які впливають на продуктивність фреймворків, є умови освітлення під час роботи з доповненою реальністю. Освітлення впливає на здатність фреймворків сканувати простір, виявляти об’єкти та забезпечувати точне розміщення віртуальних елементів. Наприклад, за слабкого освітлення шум зображення може ускладнити виявлення поверхонь, тоді як яскраве світло може викликати пересвітлення й артефакти. Для кожного типу освітлення варто створювати окремі сценарії тестування, які дозволять оцінити адаптивність алгоритмів фреймворків.

Розберемо умови та сценарії тестування на параметрі освітлення. Для інших параметрів це буде визначено у подальшій роботі.

Тож виділимо умови показника, за яких можна протестувати продуктивність фреймворків:

Тьмяне освітлення (50–100 люкс) - умови, що відповідають слабкому штучному освітленню, наприклад, у приміщенні ввечері. При такому рівні освітлення фреймворки повинні точно виявляти поверхні й об’єкти, попри високий рівень шумів у зображенні. Показники продуктивності в таких умовах оцінюються за стабільністю трекінгу та точністю створення геометрії сцени.

Денне освітлення (300–500 люкс) - звичайні денні умови у приміщенні або на вулиці без прямого сонячного світла. Це базовий сценарій для тестування, оскільки фреймворки зазвичай оптимізовані під такі умови. У таких умовах проводиться оцінка швидкості трекінгу, стабільності роботи алгоритмів і можливості виявлення дрібних деталей.

Яскраве освітлення (1000 люкс і більше) - пряме сонячне світло або інтенсивне штучне освітлення. Цей сценарій перевіряє здатність фреймворків працювати з пересвітленням, яке може спотворювати текстури та ускладнювати виявлення поверхонь. Параметри оцінки включають здатність алгоритмів адаптуватися до таких умов і мінімізувати артефакти.

Темрява (<10 люкс) - майже повна відсутність освітлення, наприклад, уночі. У таких умовах фреймворки повинні використовувати додаткові технології, такі як LiDAR, для забезпечення точного трекінгу. Показники оцінки включають точність побудови геометрії, стабільність трекінгу та швидкість оновлення сцени.

Сценарій може включати сканування кімнати площею 20 м² при рівні освітлення 50 люкс (тьмяне освітлення). У межах цього тесту оцінюються: час, необхідний для побудови сцени, точність визначення об’єктів на відстані до 3 метрів, а також стабільність трекінгу під час переміщення пристрою зі швидкістю 1 м/с. Це дозволить зрозуміти, наскільки ефективно фреймворк справляється зі складними умовами освітлення.

4.7 Проектування архітектури програмного забезпечення

Основним завданням у розробці архітектури додатку для аналізу продуктивності SceneKit та RealityKit стало створення гнучкої та масштабованої системи, яка дозволяє ефективно виконувати тестові сценарії, збирати дані та проводити їх аналіз.

У розробці використовується багаторівнева архітектура, яка складається з чотирьох основних шарів: Application Layer, Business Logic Layer, Presentation Layer. Кожен шар має чітко визначені функції та відповідає за певний аспект роботи системи. Такий підхід дозволяє розділити відповідальність між компонентами, спрощує підтримку коду та забезпечує його розширюваність (див. рис.4.1).

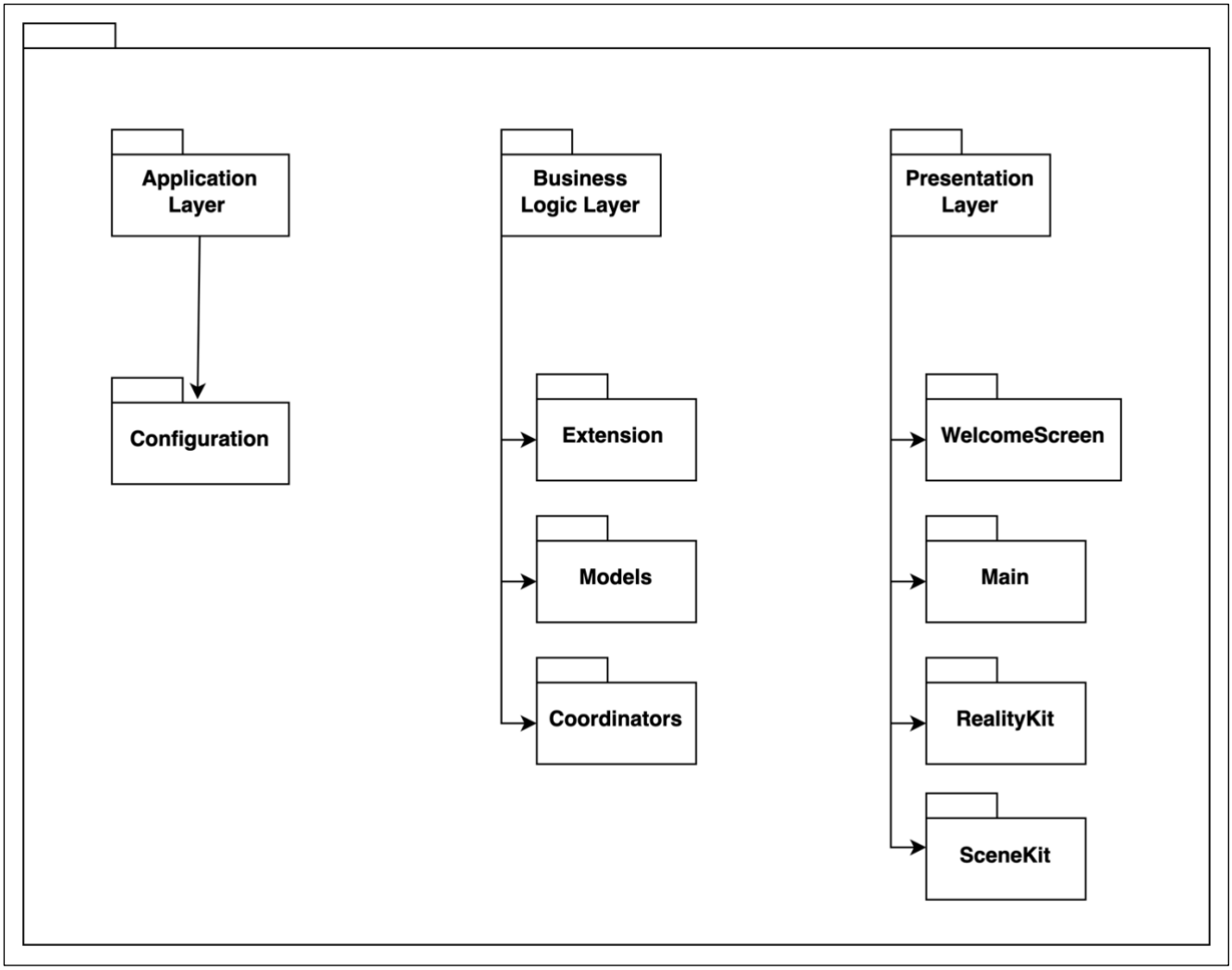


Рисунок 4.1 – Діаграма пакетів мобільного застосунка (рисунок виконаний самостійно)

Application Layer містить основні файли конфігурації додатку, системні налаштування та файли локалізації. Цей шар забезпечує базову ініціалізацію додатку, налаштування доступу до ресурсів, таких як файлові системи чи дозволи для роботи з камерою.

Business Logic Layer відповідає за реалізацію логіки тестування продуктивності. Цей шар містить координатори, які керують навігацією та організовують взаємодію між компонентами. Також тут знаходяться моделі даних, які описують основні параметри тестування, та протоколи, що визначають інтерфейси взаємодії між різними частинами системи.

Presentation Layer включає в себе презентери та представлення, які відповідають за взаємодію з користувачем. Презентери здійснюють управління відображенням даних, взаємодіючи з бізнес-логікою, і забезпечують оновлення інтерфейсу користувача відповідно до змін у додатку. Представлення (View) реалізує інтерфейс користувача та відповідає за його вигляд і поведінку.

Для реалізації додатку була обрана мова програмування Swift, яка є потужним і зручним інструментом для розробки додатків на платформі iOS. Використання фреймворків ARKit, SceneKit та RealityKit, а також бібліотек, таких як SnapKit, дозволило створити ефективний та гнучкий інструмент для аналізу. SnapKit значно полегшив процес роботи з Auto Layout, забезпечивши зручність створення адаптивного інтерфейсу для різних розмірів екранів.

Розроблена архітектура враховує можливість розширення функціональності та адаптації до нових вимог. Вона забезпечує високу стабільність, модульність і зрозумілу структуру коду, що є важливим для проведення дослідження продуктивності фреймворків і розробки рекомендацій щодо їх вибору для конкретних завдань.

5 ЕКСПЕРИМЕНТ

5.1 Визначення показників для вимірювання

Перед проведенням експерименту підсумуємо за якими критеріями будемо вимірювати продуктивність. Продуктивність AR-додатків оцінюється за допомогою кількох ключових метрик. Вони наведені в таблиці 5.1. Діапазони всіх метрик у таблиці наведені в ненормалізованому вигляді, оскільки в такому форматі їх легше сприймати.

Таблиця 5.1 – Показники продуктивності (таблиця виконана самостійно)

Показник	Діапазон	Опис
Середнє значення FPS	0-60	Визначає плавність роботи додатка
Епізодичні просідання FPS	0-60	Падіння FPS, що негативно впливає на користувацький досвід
Початкове просідання FPS	0-60	Мінімальне значення FPS під піковим навантаженням
Максимальна частота CPU, ГГц	1-3.2	Наскільки сильно процесор «розганяється» під навантаженням; побічно вказує на рівень використання CPU
Максимальне завантаження GPU, %	0-100	Відсоток використання GPU під максимальним навантаженням

Основним показником продуктивності є Frames Per Second (FPS). Висока частота кадрів забезпечує плавність роботи додатка, тоді як низька може призвести до ривків анімації та погіршення користувацького досвіду. Для AR/VR-додатків оптимальним значенням вважається стабільні 60 FPS. Вимірювання FPS дозволяє оцінити, наскільки ефективно фреймворки обробляють рендеринг сцени та підтримують стабільну продуктивність під час анімацій.

Використання CPU є критичним для загальної продуктивності додатка. Процесор відповідає за завантаження ресурсів, управління 3D-об’єктами, передачу даних у GPU, обробку фізичних взаємодій, інтеграцію даних із сенсорів тощо. Ефективне використання CPU необхідне для досягнення балансу між плавністю анімації та мінімальним споживанням ресурсів.

Мінімальне значення частоти CPU встановлено на рівні 1 ГГц, оскільки це відповідає його режиму очікування. Верхня межа встановлена на рівні 3.2 ГГц, оскільки у більшості iPhone з чіпами серії AX частота «великих» ядер може досягати 2.3-3.2 ГГц.

Використання GPU відіграє ключову роль у досягненні високої якості графіки та інтерактивності в AR/VR-додатках. GPU спеціалізується на рендерингу 3D-об’єктів, текстур, тіней, віддзеркалень та освітлення. У AR-додатках GPU забезпечує коректну інтеграцію віртуальних об'єктів у зображення з камери. Високі обчислювальні навантаження можуть спричинити падіння FPS і негативно вплинути на користувацький досвід, особливо у складних сценах із великою кількістю об’єктів. Для підтримки 60 FPS або вище продуктивність GPU має бути оптимізована, щоб уникнути затримок рендерингу, уповільнення інтерфейсу та надмірного споживання батареї.

Перевантаження GPU може призвести до перегріву та швидкої розрядки пристрою, що робить важливим збалансоване управління ресурсами для ефективної роботи AR-додатків.

Адаптивність фреймворків дозволяє розробникам оптимізувати використання ресурсів із врахуванням обмежень пристроїв. Ефективне управління CPU і GPU допомагає уникнути надмірного зниження продуктивності та покращує загальний користувацький досвід.

Для проведення MCDA фреймворків SceneKit та RealityKit у контексті практичного використання в AR-додатках було вирішено узагальнити наявні детальні критерії, виявлені у п.4 та згрупувати їх за зручними і репрезентативними групами, що відображають ключові аспекти розробки. Це дозволяє отримати цілісне уявлення про функціональні можливості, обмеження та сильні сторони

кожного з фреймворків, а також виконати подальшу оцінку доцільності їх використання для реалізації прикладних сценаріїв.

Надані згруповані критерії та їх усереднені підсумкові оцінки для кожної групи можна знайти в таблиці 5.2.

Таблиця 5.2 – Інтегровані критерії (таблиця виконана самостійно)

Груповий критерій	Внутрішні критерії	SceneKit (усереднене значення)	RealityKit (усереднене значення)
Графічні можливості	формати 3D-файлів, анімації, фотореалізм, фізичні симуляції,	7.5	7.25
Налаштування та кастомізація	оптимізація продуктивності, кастомізація	8.5	5
Обмеження фреймворків	сумісність, масштабованість складних сцен	8.5	9
Простота використання	спільнота/документація, структура та побудова сцени, необхідність оптимізації	7.5	8.3

Це дозволило структурувати порівняльний аналіз за узагальненими критеріями, що є необхідним етапом для подальшого багатокритеріального ранжування в рамках MCDA.

Окремо треба оцінити критерій ступінь інтеграції функцій AR, оскільки його не доцільно відносити ні до якої групи. За цим показником SceneKit отримав 6 балів, тоді як RealityKit – 9 балів. Це пов'язано з тим, що SK підтримує лише базові функції AR Anchoring та освітлення навколишнього середовища з коробки. Інші функції, такі як виявлення площини в реальному часі, оклюзія в реальному часі, взаємодія з реальним світом на основі фізики, підтримка захоплення руху тощо, підтримуються за допомогою ручного налаштування за допомогою ARKit або не підтримуються взагалі.

5.2 Сценарії тестування продуктивності

Для проведення експерименту було розроблено iOS-застосунок ARPerfTester, який дозволяє аналізувати продуктивність SceneKit і RealityKit у рендерингу сцен, використанні ресурсів CPU та GPU, а також виконанні однакових операцій. Цей застосунок надає користувачеві можливість створювати тривимірну сцену, додавати й видаляти об'єкти, виконувати анімації та аналізувати поведінку системи за різного рівня навантаження.

Для аналізу ефективності роботи SceneKit та RealityKit було розроблено три основні сценарії тестування, які охоплюють ключові аспекти рендерингу, обробки анімацій та управління пам’яттю.

Вибір тестових сценаріїв зумовлений необхідністю оцінки продуктивності фреймворків у реальних умовах роботи AR-застосунків, з урахуванням можливого навантаження на апаратні ресурси пристрою.

Сценарій тестування 1: вплив кількості об'єктів на продуктивність рендерингу (див. рис.5.1 та рис.5.2).

```
func addNode() {
  let cubeSize = CGFloat.random(in: 0.5...1.5)
  let cube = SCNBox(width: cubeSize, height: cubeSize, length: cubeSize, chamferRadius:
    0.0)

  let material = SCNMaterial()
  material.diffuse.contents = UIColor.random()
  cube.materials = [material]

  let cubeNode = SCNNode(geometry: cube)
  let randomX = Float.random(in: -5...5)
  let randomY = Float.random(in: -5...5)
  let randomZ = Float.random(in: -5...5)
  cubeNode.position = SCNVector3(randomX, randomY, randomZ)

  selfView.scene?.rootNode.addChildNode(cubeNode)
  nodes.append(cubeNode)
  delegate?.nodesWereUpdated(nodes: nodes, from: self)
}
```

Рисунок 5.1 – SceneKit. Фрагмент коду додавання об'єктів у сцену (рисунок виконаний самостійно)

Даний тестовий сценарій має на меті визначити, як SceneKit і RealityKit справляються із завданням рендерингу великої кількості об'єктів у сцені. Для цього було поступово збільшено кількість об'єктів від 10 до 100, загальна кількість

полігонів при цьому досягала 10 000. Основний параметр оцінки – частота кадрів (FPS), що є критичним показником плавності роботи AR-застосунків. Також фіксувалися пікові навантаження на процесор (CPU) та графічний процесор (GPU).

```
func addNode() {
  let cubeSize = CGFloat.random(in: 0.5...1.5)
  let cube = SCNBox(width: cubeSize, height: cubeSize, length: cubeSize, chamferRadius:
    0.0)

  let material = SCNMaterial()
  material.diffuse.contents = UIColor.random()
  cube.materials = [material]

  let cubeNode = SCNNode(geometry: cube)
  let randomX = Float.random(in: -5...5)
  let randomY = Float.random(in: -5...5)
  let randomZ = Float.random(in: -5...5)
  cubeNode.position = SCNVector3(randomX, randomY, randomZ)

  selfView.scene?.rootNode.addChildNode(cubeNode)
  nodes.append(cubeNode)
  delegate?.nodesWereUpdated(nodes: nodes, from: self)
}
```

Рисунок 5.2 – RealityKit. Фрагмент коду додавання об'єктів у сцену (рисунок виконаний самостійно)

Аналіз даних дозволяє оцінити ефективність обробки графічних об'єктів, оптимізацію роботи фреймворків та їхню здатність підтримувати стабільну продуктивність при збільшенні навантаження.

Сценарій тестування 2: вплив анімацій на продуктивність (див. рис. 5.3 та рис.5.4).

```
func startAnimation() {
  for cube in nodes {
    let rotateAction = SCNAction.rotateBy(x: 0, y: CGFloat.pi, z: 0, duration: 1)
    let repeatAction = SCNAction.repeatForever(rotateAction)
    cube.runAction(repeatAction)
  }
}

func stopAnimation() {
  for cube in nodes {
    cube.removeAllActions()
  }
}
```

Рисунок 5.3 – SceneKit. Фрагмент коду реалізації анімації об'єкта (рисунок виконаний самостійно)

```
func rotateEntity(_ entity: Entity) {
  let currentPosition = entity.transform.translation
  let from = Transform(scale: .one, rotation: .init(angle: 0, axis: [0, 1, 0]),
    translation: currentPosition)
  let to = Transform(scale: .one, rotation: .init(angle: .pi, axis: [0, 1, 0]),
    translation: currentPosition)
  let definition = FromToByAnimation(from: from,
                                     to: to,
                                     duration: 1,
                                     timing: .default,
                                     bindTarget: .transform,
                                     repeatMode: .repeat)

  if let animation = try? AnimationResource.generate(with: definition) {
    entity.playAnimation(animation)
  }
}

func startRotationAnimation() {
  for entity in entities {
    rotateEntity(entity)
  }
}

func stopAnimation() {
  for entity in entities {
    entity.stopAllAnimations()
  }
}
```

Рисунок 5.4 – RealityKit. Фрагмент коду реалізації анімації об'єкта (рисунок виконаний самостійно)

Оскільки анімації є важливим елементом AR-застосунків, було досліджено, як SceneKit та RealityKit обробляють одночасне виконання анімацій для великої кількості об'єктів. Для цього було створено сцену з об'єктами (до 100 об'єктів, до 10 000 полігонів), на які одночасно накладалася анімація обертання. У процесі тестування оцінювалися показники FPS, використання CPU та GPU, а також наявність просідань продуктивності під час виконання анімацій. Важливим аспектом аналізу було визначення рівня оптимізації роботи анімаційного механізму фреймворків та їхньої здатності підтримувати стабільну частоту кадрів без значного зростання навантаження на апаратні ресурси.

Сценарій тестування 3: оцінка стабільності після видалення об'єктів (див. рис.5.5 та рис.5.6).

Цей сценарій спрямований на дослідження механізмів управління пам’яттю та вивільнення ресурсів у SceneKit і RealityKit. Після додавання та анімації об'єктів у сцені здійснювалося їхнє поступове видалення. Аналізувалися такі параметри, як швидкість вивільнення пам’яті, вплив цього процесу на загальну продуктивність та

стабільність FPS після видалення об'єктів. Особлива увага приділялася тому, чи залишаються «завислі» об'єкти в пам’яті після їхнього видалення та чи впливає цей фактор на подальшу роботу застосунку.

```
func removeLastNode() {
  guard let lastCube = nodes.popLast() else { return }
  lastCube.removeFromParentNode()
  delegate?.nodesWereUpdated(nodes: nodes, from: self)
}
```

Рисунок 5.5 – SceneKit. Фрагмент коду видалення об'єктів у сцену (рисунок виконаний самостійно)

```
func removeLastNode() {
  guard let lastEntity = entities.popLast() else { return }
  lastEntity.removeFromParent()
  delegate?.entitiesWereUpdated(entities: entities, from: self)
}
```

Рисунок 5.6 – RealityKit. Фрагмент коду видалення об'єктів у сцену (рисунок виконаний самостійно)

Результати тестування за цими трьома сценаріями дозволяють отримати об’єктивну оцінку продуктивності фреймворків, їхньої здатності ефективно керувати графічними ресурсами та підтримувати стабільну роботу навіть за умов високого навантаження.

Подальший аналіз експериментальних даних допоможе визначити переваги та обмеження кожного фреймворку, що має критичне значення при виборі відповідного інструменту для розробки AR-застосунків.

5.3 Вимірювання продуктивності

Для аналізу продуктивності було використано Xcode Instruments [18] – основний інструмент для профілювання iOS-застосунків. Instruments дозволяє в режимі реального часу відстежувати використання ресурсів пристрою, включаючи CPU, GPU, FPS, пам’ять і енергоспоживання. Для повноцінного аналізу

продуктивності фреймворків SceneKit та RealityKit було здійснено детальне налаштування інструментів профілювання в середовищі Xcode Instruments. Рисунки 5.7-5.10 ілюструють налаштування тих чи інших іструментів або відображення збору відповідних даних під час виконання сцен.

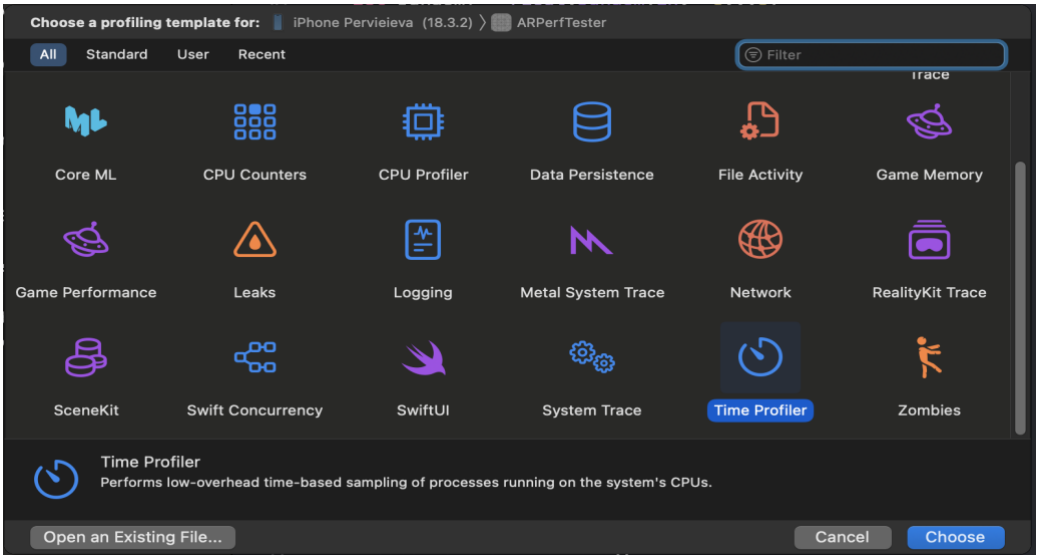


Рисунок 5.7 – Скриншот додавання Time Profiler інструменту (рисунок виконаний самостійно)

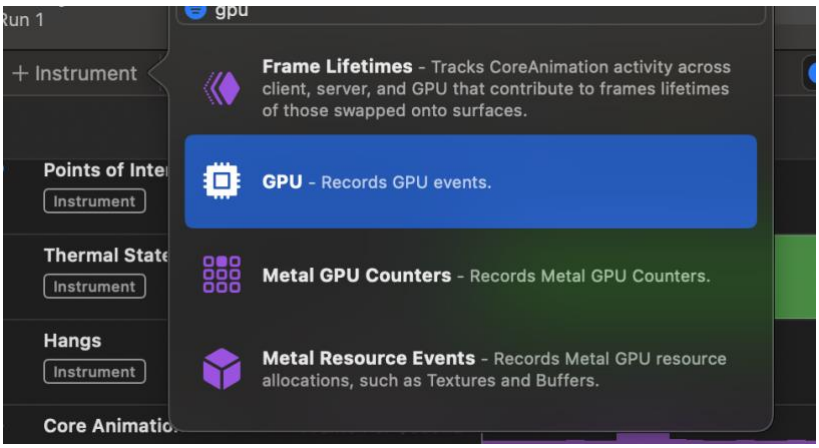


Рисунок 5.8 – Скриншот додавання GPU Profiler інструменту (рисунок виконаний самостійно)

Перед запуском тестів у Xcode Instruments було налаштовано такі інструменти:

Time Profiler – для аналізу завантаження CPU і визначення пікових моментів обробки сцени;

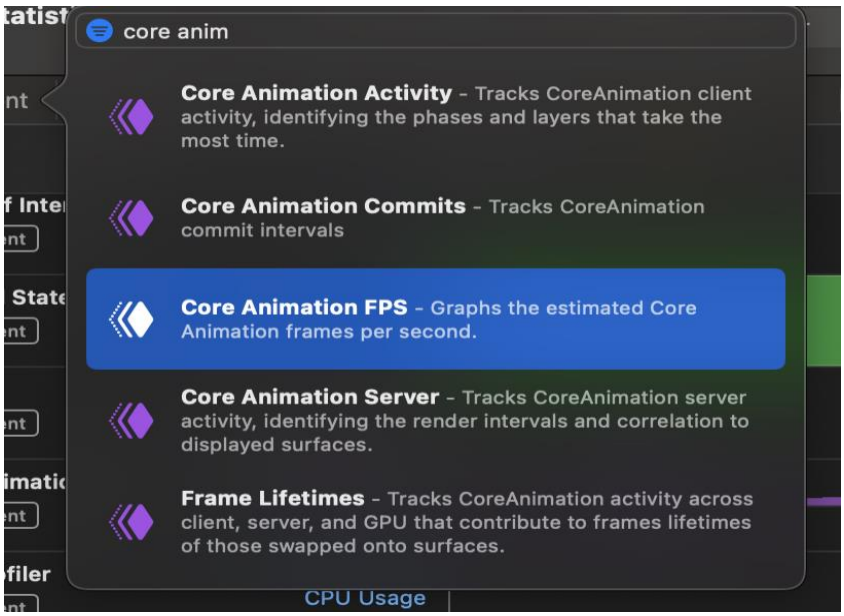


Рисунок 5.9 – Скриншот додавання Core Animation FPS інструменту (рисунок виконаний самостійно)

Рисунок 5.10 – Інтерфейс Xcode Instruments файл .trace (рисунок виконаний самостійно)

GPU Profiler – для моніторингу використання GPU під час рендерингу 3D-об'єктів та анімацій;

Core Animation – для відстеження частоти кадрів (FPS) у реальному часі та оцінки стабільності фреймворків.

Основні тести проводились на фізичному пристрої iPhone 15 Pro з 8 ГБ оперативної пам’яті під управлінням iOS 18. Під час кожного тестового сценарію

Xcode Instruments реєстрував усі показники продуктивності, дозволяючи як аналіз у реальному часі, так і подальший перегляд результатів. Зібрані дані експортувалися у формат .trace, що дозволило здійснити детальний подальший аналіз показників.

Ці візуальні приклади допомагають краще зрозуміти принципи роботи з інструментами продуктивності в iOS-розробці та підтверджують, що всі заміри виконувалися практично й на реальному пристрої.

Отримані результати було зафіксовано у вигляді чисельних показників продуктивності за допомогою .trace-файлів. Нижче наведено узагальнену таблицю з ключовими результатами експерименту (див. табл. 5.3).

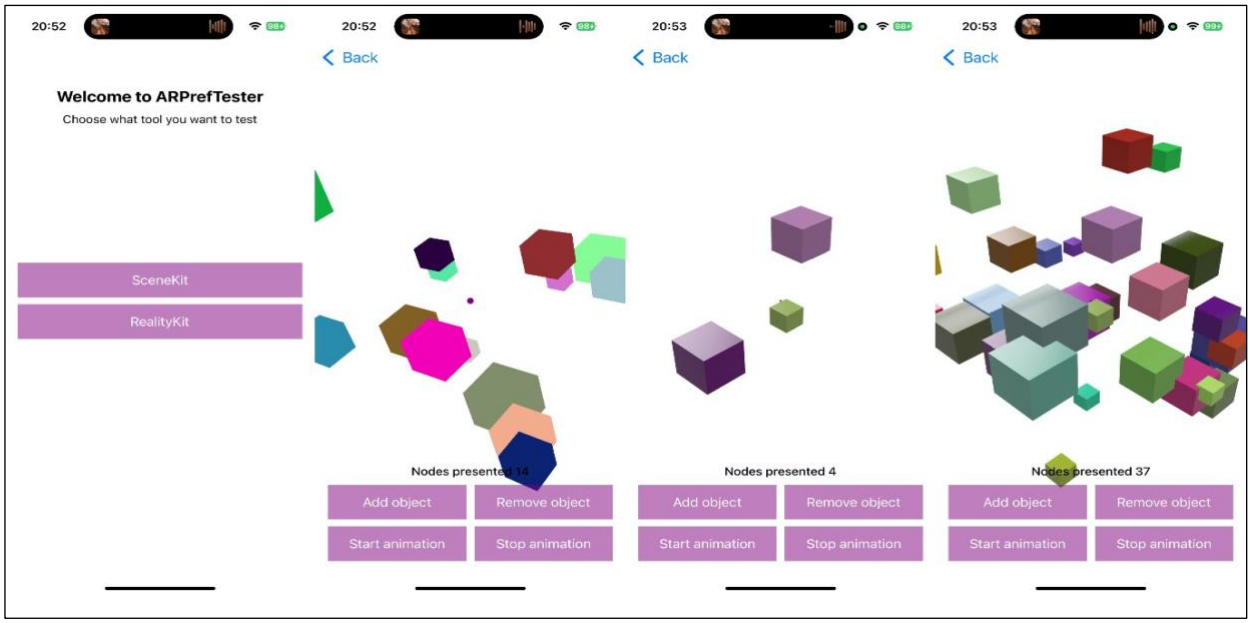
Таблиця 5.3 – Підсумкові результати експерименту (таблиця виконана самостійно)

Показник	SceneKit	RealityKit
Середнє значення FPS	58	54
Епізодичні просідання FPS	4	9
Початкове просідання FPS	55	9
Максимальна частота CPU, ГГц	1.68	2.13
Максимальне завантаження GPU, %	18	53

Як видно з результатів, SceneKit демонструє стабільнішу частоту кадрів та менше просідань у критичних моментах, водночас RealityKit характеризується вищим навантаженням на GPU і CPU, що дозволяє досягти кращої графіки, але потребує більшої кількості апаратних ресурсів.

На рисунку 5.11 продемонстровано вигляд сцени на реальному пристрої під час виконання тестових сценаріїв. Візуалізації охоплюють ключові етапи тестування, перший екран – SceneKit, останні – RealityKit. Вони ілюструють, як саме відображались об'єкти на сцені, як виконувалась анімація, а також яким було загальне візуальне навантаження у процесі експерименту. Це дозволяє не лише

зафіксувати результати вимірів, але й порівняти графічну реалізацію та поведінку сцен у різних фреймворках в умовах однакових сценаріїв.



а) б) в) г)

Рисунок 5.11 – Вигляд сцени а) обрання фреймворку; б) SceneKit, тест1; в)RealityKit,тест 2; г) RealityKit,тест 3 (рисунок виконаний самостійно)

Детальний порівняльний аналіз результатів наведено у наступному розділі.

5.4 Аналіз результатів

Протягом більшої частини тестування SceneKit демонстрував високу продуктивність, проте короткочасні падіння FPS спостерігалися під час додавання великої кількості об'єктів. Ці просідання відбувалися на початковому етапі змін сцени, коли система обробляла великий масив нових об'єктів або запускала одночасну анімацію.

RealityKit мав більш виражені коливання FPS на початковому етапі (коли сцена була ще порожньою) – частота кадрів знижувалася до 9 FPS, що, ймовірно, пов’язано із високими обчислювальними вимогами до фізичних ефектів і початкового рендерингу. Після цього FPS стабілізувався в межах 57-60 FPS, але періодичні просідання продуктивності все ще спостерігалися. Вони не були такими

глибокими, як у SceneKit, але RealityKit стабілізувався довше, ніж SceneKit. Це підтверджує вищі вимоги RK до системних ресурсів.

Завантаження GPU у SceneKit залишалося відносно низьким, що свідчить про ефективну оптимізацію обчислень, де навантаження рівномірно розподіляється між процесами. У RealityKit використання GPU було значно вищим, особливо під час активної взаємодії з об'єктами, коли рендеринг складного освітлення, фізичних ефектів та анімацій викликав пікове навантаження.

Частота CPU в SceneKit зростала під час додавання нових об'єктів і виконання анімацій, але залишалася стабільною після видалення об'єктів. RealityKit мав значно вищі обчислювальні вимоги, ніж SceneKit, що пояснюється широким використанням фізичних симуляцій і освітлення, які підвищують навантаження на процесор.

Результати експерименту показали, що SceneKit є більш стабільним у підтримці високої частоти кадрів та ефективно використовує апаратні ресурси, особливо GPU, що забезпечує кращу автономність пристрою та оптимізовану продуктивність у випадках обмежених ресурсів.

З іншого боку, RealityKit забезпечує вищу якість рендерингу, складніші фізичні взаємодії та фотореалізм, але це супроводжується значно вищим навантаженням на CPU і GPU, що може призвести до нестабільності FPS та підвищеного енергоспоживання.

Для подальшого аналізу отримані значення були нормалізовані до діапазону [0-10] за формулою 5.1:

$$x_{norm} = \frac{x_i - x_{min}}{x_{max} - x_{min}} * 10 \quad (5.1)$$

Результати нормалізації наведені в таблиці 5.4.

Таблиця 5.4 – Нормалізовані результати експерименту (таблиця виконана самостійно)

Показник	SceneKit	RealityKit
Середнє значення FPS	9.7	9
Епізодичні просідання FPS	0.67	1.5
Початкове просідання FPS	9.17	1.5
Максимальна частота CPU, ГГц	3.09	5.14
Максимальне завантаження GPU, %	1.8	5.3

5.5 Застосування багатокритеріального аналізу для оцінки фреймворків

Отже, для порівняння фреймворків SceneKit та RealityKit за допомогою MCDA пропонується використовувати критерії, перелік яких наведено у таблиці 5.5. Проведене узагальнення дозволяє наочно побачити, що SceneKit демонструє сильні позиції в можливостях кастомізації та продуктивності під час складних сценаріїв, але поступається RealityKit у простоті використання та інтеграції з AR. RealityKit же, навпаки, більше підходить для швидкої розробки додатків із високим ступенем інтерактивності та реалістичності без потреби глибоких налаштувань. Така система критеріїв буде використовуватися далі у розділі аналізу прикладних кейсів.

Таблиця 5.5 – Зведена порівняльна оцінка фреймворків (таблиця виконана самостійно)

Критерій	SceneKit	RealityKit
Продуктивність	4.89	4.49
Графічні можливості	7.5	7.25
Інтеграція з AR	6	9
Налаштування та кастомізація	8.5	5
Простота використання	7.5	8.3
Обмеження	8.5	9

Для оцінки доцільності використання фреймворків ARKit та RealityKit у різних умовах були розглянуті два типові сценарії застосування. Використано метод MCDA, який дозволяє об’єктивно порівняти технології на основі вагових коефіцієнтів для різних груп показників продуктивності та функціональності.

Застосунок 1 розробляється невеликою командою стартапу. Це каталог продукції, що містить меблі, а також утиліта для розміщення 3D-моделей у реальному середовищі. Основними вимогами до такого додатка є висока якість візуалізації, можливість взаємодії з об’єктами через жести, коректне врахування перспективи та світлових умов, а також простота використання. У такому контексті ключову роль відіграють можливості роботи з AR, наявність фотореалістичного рендерингу та інтеграція з ARKit для точного розміщення об'єктів.

Застосунок 2 розробляється командою, що має досвід у 3D-графіці або розробці ігор. Це освітня фізична лабораторія, яка дозволяє студентам розміщувати віртуальні експериментальні установки (маятники, механічні моделі тощо) в реальному середовищі. Додаток дає змогу досліджувати та експериментувати з точними фізичними взаємодіями, налаштовуючи такі характеристики, як крутний момент двигуна, тертя в з'єднаннях, передаточні відношення шестерень, обмеження кутів, розподіл маси, демпфування, міцність з'єднань тощо. У цьому випадку критичними факторами є точність фізичного моделювання, можливість детального налаштування взаємодії об'єктів та продуктивність під час опрацювання складних графічних об'єктів.

Вагові коефіцієнти для кожного типу додатку були обчислені з використанням методики AHP (аналітичний ієрархічний процес) [17] з припущенням, що всі експерти рівноправні. Це дозволило забезпечити більш об’єктивну та прикладну оцінку можливостей фреймворків у реальних умовах розробки. Використання MCDA дозволяє оцінити відповідність фреймворків до поставлених вимог. Виконано адитивну згортку показників для кожного фреймворку та сценарію. Результати оцінки для першого застосунку наведені в таблиці 5.6.

Таблиця 5.6 – Порівняння SceneKit та RealityKit для каталогу меблів (таблиця виконана самостійно)

Критерій	Вага	SceneKit		RealityKit	
		c_i	$c_i * w_i$	c_i	$c_i * w_i$
Продуктивність	0,25	4.89	1.22	4.49	1.12
Графічні можливості	0,1	7.5	0.75	7.25	0.725
Інтеграція з AR	0,25	6	1.5	9	2.25
Налаштування та кастомізація	0,1	8.5	0.85	5	0.5
Простота використання	0,2	7.5	1.5	8.2	1.7
Обмеження	0,1	8.5	0.85	9	0.9
Загальна оцінка			6.67		7.2

Експериментальні результати демонструють, що для реалізації першого застосунку – мобільного каталогу меблів із можливістю розміщення 3D-моделей у просторі – більш придатним є фреймворк RealityKit. Це обумовлено його потужними можливостями у сфері AR-взаємодії, зручністю інтеграції, високим рівнем фотореалізму «з коробки» та мінімальними вимогами до налаштування.

Результати оцінки для другого застосунку наведені в таблиці 5.7.

Таблиця 5.7 – Порівняння SceneKit та RealityKit для технічного застосунку (таблиця виконана самостійно).

Критерій	Вага	SceneKit		RealityKit	
		c_i	$c_i * w_i$	c_i	$c_i * w_i$
1	2	3	4	5	6
Продуктивність	0,25	4.89	1.22	4.49	1.12
Графічні	0,2	7.5	1.5	7.25	2.18

Кінець таблиці 5.7

1	2	3	4	5	6
можливості					
Інтеграція з AR	0,15	6	0.9	9	1.35
Налаштування та кастомізація	0,3	8.5	1.7	5	1
Простота використання	0,05	6.7	0.38	8.5	0.43
Обмеження	0,05	8.5	0.43	9	0.45
Загальна оцінка			6.88		6.53

Для другого застосунку – віртуальної фізичної лабораторії для навчальних цілей, де користувач повинен мати змогу налаштовувати фізичні властивості об'єктів (інерцію, тертя, масу тощо), виконувати складні симуляції та отримувати точні результати в реальному часі – більш придатним виявився фреймворк SceneKit. Це пояснюється його підтримкою гнучкого налаштування, можливістю реалізації кастомних шейдерів, широкими інструментами для оптимізації продуктивності та точнішою підтримкою складної графіки, яка необхідна для фізичних моделей.

Вибір фреймворку залежить не лише від базової продуктивності чи простоти використання, а насамперед від специфіки функціональності, яка визначає вимоги до візуалізації, взаємодії, фізики та кастомізації в кожному конкретному сценарії.

Результати аналізу показують, що RealityKit є кращим вибором для простих AR-застосунків, які потребують швидкої розробки та легкої інтеграції з ARKit. Фреймворк добре підходить для реалізації таких функцій, як розміщення 3D-об’єктів у просторі, інтеграція з камерою та взаємодія з користувачем без необхідності складної настройки сцени та графічного рендерингу.

SceneKit, у свою чергу, є оптимальним вибором для технічних застосунків, що потребують кастомізації графіки, складних візуальних ефектів та точного

керування рендерингом. Його можливості дозволяють створювати індивідуальні рішення з використанням шейдерів, кастомних текстур і складних анімацій, що важливо у спеціалізованих AR-додатках.

Таким чином, вибір між цими фреймворками залежить від специфіки задачі:

- RealityKit підходить для інтерактивних AR-додатків з мінімальними вимогами до складної графіки;
- SceneKit – кращий вибір для візуалізації з високим рівнем кастомізації та складною графічною логікою.

Отримані результати дозволяють сформувати рекомендації для розробників AR-додатків, що допоможе обрати оптимальний інструмент для конкретних проектів з урахуванням їхніх вимог.

Вихідний код розробленого застосунку доступний для завантаження з [19]. Основні положення роботи були представленні у статті доповіді [20].

ВИСНОВКИ

У цій роботі було проведено комплексне дослідження фреймворків для розробки додатків доповненої реальності на платформі iOS.

На початковому етапі роботи було проведено огляд літератури, що охоплювала основні аспекти використання фреймворків ARKit, SceneKit та RealityKit. Огляд літератури дозволив ідентифікувати сильні та слабкі сторони кожного фреймворку та виявити їхню актуальність у розробці сучасних AR-додатків.

На основі отриманих даних було виконано теоретичне дослідження переваг і недоліків кожного з фреймворків. ARKit продемонстрував свою ефективність у точному трекінгу та інтеграції з реальним простором, SceneKit – у створенні та рендерингу 3D-об’єктів, а RealityKit – у забезпеченні фотореалістичності й інтерактивності. Однак було визначено, що зв’язка ARKit + SceneKit має найбільший потенціал для розробки комплексних AR-додатків, де потрібно поєднати можливості трекінгу та роботи з графікою. Такий підхід дозволяє компенсувати обмеження кожного з фреймворків, створюючи більш гнучке та ефективне рішення.

Для об’єктивної оцінки фреймворків було виділено ключові критерії, які поділено на кількісні та якісні. Кількісні критерії включають метрики продуктивності, такі як FPS, навантаження на CPU та GPU, час завантаження сцен і об’єктів. Якісні критерії охоплюють підтримку форматів 3D-моделей, можливості VR/AR операцій, побудову сцен, освітлення, анімації та легкість використання.

Для якісних критеріїв було розроблено чіткі шкали оцінювання, які дозволяють об’єктивно аналізувати можливості фреймворків. Наприклад, побудова сцен оцінювалася за рівнем деталізації, кількістю об’єктів і складністю інтерактивності. Освітлення враховувало типи джерел світла, реалістичність тіней і адаптивність до змін умов. Легкість використання оцінювалася за доступністю документації, зручністю API та підтримкою інтеграції.

На основі формалізованих критеріїв було проведено аналіз фреймворків. Результати дослідження було представлено у вигляді таблиць, що демонструють

порівняння фреймворків за кожним критерієм. Наприклад, RealityKit отримав найвищі оцінки за підтримку фотореалістичного освітлення та інтеграцію з фізичними симуляціями, тоді як зв’язка ARKit + SceneKit показала кращі результати у підтримці широкого спектру форматів 3D-моделей. Ці результати дали змогу зрозуміти сильні й слабкі сторони кожного підходу.

У ході роботи було виділено показники, які суттєво впливають на продуктивність фреймворків у реальних сценаріях використання. До них відносяться складність сцени (кількість полігонів і об’єктів), динамічність анімацій та освітлення за різних умов (тускле, денне, нічне). У подальшій роботі будуть виділені конкретні сценарії використання, за якими оцінюватимуться можливості та продуктивність фреймворків у MCDA.

У цьому дослідженні було розроблено та реалізовано спеціальний застосунок ARPerfTester для проведення експериментального порівняння продуктивності фреймворків SceneKit та RealityKit у розробці AR-застосунків для iOS. Було розроблено набір тестових сценаріїв, які дозволили оцінити роботу обох фреймворків у різних режимах навантаження, зокрема при збільшенні кількості об'єктів, одночасному виконанні анімацій та видаленні елементів сцени. За результатами експерименту було отримано кількісні показники продуктивності: середнє значення FPS, епізодичні та початкові просідання FPS, максимальне завантаження CPU та GPU.

Отримані дані стали основою для проведення MCDA, що дозволив об’єктивно порівняти SceneKit і RealityKit за ключовими параметрами. В рамках MCDA були виділені основні групи критеріїв: продуктивність, графічні можливості, інтеграція з AR, налаштування та кастомізація, простота використання та обмеження. На основі цих критеріїв було визначено переваги кожного з фреймворків залежно від конкретних завдань проєкту.

За результатами експерименту рекомендовано використовувати RealityKit для проєктів, де важливими є візуальна складність, реалістичність та зручність створення інтерактивних сцен із мінімальними зусиллями з боку розробника. Водночас SceneKit є оптимальним вибором для проєктів, які потребують гнучкого

налаштування, глибокої кастомізації графіки та підтримки специфічних ефектів, котрі неможливо реалізувати засобами RealityKit без додаткових рішень.

Особливо важливо враховувати, що під час використання MCDA у проєкті можуть існувати обов’язкові критерії – наприклад, підтримка власних шейдерів або гнучке управління постобробкою. Такі критерії слід перевіряти перед початком MCDA, а альтернативи, які їм не відповідають, необхідно виключати з розгляду. Саме порівняння за допомогою MCDA доцільно проводити лише для тих рішень, які пройшли попередній відбір за ключовими вимогами.

У перспективі дослідження варто зосередитися на аналізі інтеграції AI-засобів у AR-середовище, таких як семантичне розпізнавання сцен, покращене трекінгове позиціювання та автоматична класифікація об'єктів. Це дозволить суттєво підвищити функціональність нативних iOS-застосунків та розширити можливості фреймворків у майбутніх розробках.

Результати виконаного дослідження були апробовані на «XVIII Міжнародній науково-практичній конференції магістрантів та аспірантів «Теоретичні та практичні дослідження молодих вчених», 19-22 листопада 2024 р.»(див. додаток В). Також за результатами дослідження було підготовлено та прийнято матеріали на міжнародну конференцію «16th International Scientific and Practical Conference «Environment. Technology. Resources», 19-20 червня, 2025р.» (див. додаток Г).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Доповнена реальність. Apple (Україна). URL: https://www.apple.com/ua/augmented-reality/ (дата звернення: 20.05.2025).

2. Первєєва Д. О., Широкопетлєва М. С. Порівняння характеристик інструментів для роботи з доповненою реальністю на iOS // XVIII Міжнародна науково-практична конференція магістрантів та аспірантів «Теоретичні та практичні дослідження молодих вчених» (19-22 листопада 2024 року): матеріали конференції / за ред. проф. Є. І. Сокола. – Харків: НТУ «ХПІ», 2024. – С. 193-194.

3. Nowacki P., Woda M. S. Capabilities of ARCore and ARKit Platforms for AR/VR Applications // Engineering in Dependability of Computer Systems and Networks. – 2020. – № 987. – С. 358-370.

4. Sure T. A. R. Motion Tracking in iOS Applications Using Augmented Reality // Journal of Android and IOS Applications and Testing. – 2023. – С. 1-5.

5. Nissen L., et al. Towards Preventing Gaps in Health Care Systems through Smartphone Use: Analysis of ARKit for Accurate Measurement of Facial Distances in Different Angles // Sensors. – 2023. – Т. 23, № 9 (4486). – С. 1-18. URL: https://doi.org/10.3390/s23094486 (дата звернення: 20.05.2025).

6. Nhân J. Building Your First ARKit App with SceneKit // Mastering ARKit: Apple’s Augmented Reality App Development Platform. – Berkeley, CA: Apress, 2022. – С. 59–78. URL: https://doi.org/10.1007/978-1-4842-7836-9_4

7. Gaol F. L., et al. Augmented Reality SDK Overview for General Application Use // International Journal of Advanced Computer Science & Applications. – 2023. – Т. 14, № 11. – С.54-60.

8. Cabadajová M. Comparison of ARKit, ARCore and Supported Rendering Technologies in iOS Development: Master’s Thesis / Masaryk University, Faculty of Informatics. – Brno, 2023. – 81 с.

9. Gruzdo I., Kyrychenko I., Tereshchenko G., Shanidze O. Analysis of Models Usability Methods Used on Design Stage to Increase Site Optimization // CEUR Workshop Proceedings. – 2023. – № 3403. – С. 387-409.

10. ARKit. Apple Developer Documentation. Apple Developer Documentation. URL: https://developer.apple.com/documentation/arkit (дата звернення: 20.05.2025).

11. SceneKit. Apple Developer Documentation. Apple Developer Documentation. URL: https://developer.apple.com/documentation/scenekit (дата звернення: 20.05.2025).

12. RealityKit. Apple Developer Documentation. Apple Developer Documentation. URL: https://developer.apple.com/documentation/realitykit (дата звернення: 20.05.2025).

13. Boutsi A.-M., Ioannidis C., Verykokou S. Multi-Resolution 3D Rendering for High-Performance Web AR // Sensors. – 2023. – Т. 23, № 15(6885). – С.1-25. URL: https://doi.org/10.3390/s23156885 (дата звернення: 20.05.2025).

14. Hort M., Kechagia M., Sarro F., Harman M. A Survey of Performance Optimization for Mobile Applications // IEEE Transactions on Software Engineering. – 2022. – Т. 48, № 8. – С. 2879-2904. DOI: 10.1109/TSE.2021.3071193.

15. Nusrat F., Hassan F., Zhong H., Wang X. How Developers Optimize Virtual Reality Applications: A Study of Optimization Commits in Open Source Unity Projects // IEEE/ACM 43rd International Conference on Software Engineering (ICSE). – Madrid, ES, 2021. – С. 473-485. DOI: 10.1109/ICSE43902.2021.00052.

16. Gruzdo I., Kyrychenko I., Tereshchenko G., Shanidze N. Metrics Applicable for Evaluating Software at the Design Stage // 5th International Conference on Computational Linguistics and Intelligent Systems (COLINS-2021), April 22–23, 2021, Kharkiv, Ukraine. CEUR Workshop Proceedings. – 2021. – Т. 2870, Вип. I. – С. 916-936.

17. Ishizaka A., Nemery P. Multi-Criteria Decision Analysis: Methods and Software. – Hoboken, NJ: Wiley, 2013. – 296 с. DOI: 10.1002/9781118644898.

18. Getting Started with Instruments. WWDC Notes. URL: https://wwdcnotes.com/documentation/wwdcnotes/wwdc19-411-getting-started-with-instruments/ (дата звернення: 20.05.2025).

19. GitHub – ARPerfTester. *GitHub*. URL: https://github.com/Daria-Koshkina/ARPerfTester (дата звернення: 20.05.2025).

20. Shirokopetleva M., Shevchenko O., Pervieieva D., Savulioniene L., Sakalys P. A Comparative Analysis of Native iOS Frameworks for Developing Graphics Scenes in Augmented Reality Applications // Environment. Technology. Resources. Proceedings of the 16th International Scientific and Practical Conference – Rezekne, Latvia, 2025. – Vol. II. – C. 321–327. URL: https://doi.org/10.17770/etr2025vol2.8610

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ ЗА НАУКОВИМИ НАПРЯМАМИ КЕРІВНИКА ТА НАУКОВЦІВ КАФЕДРИ ПРОГРАМНОЇ ІНЖЕНЕРІЇ

2. Первєєва Д. О., Широкопетлєва М. С. Порівняння характеристик інструментів для роботи з доповненою реальністю на iOS // XVIII Міжнародна науково-практична конференція магістрантів та аспірантів «Теоретичні та практичні дослідження молодих вчених» (19–22 листопада 2024 року): матеріали конференції / за ред. проф. Є. І. Сокола. – Харків: НТУ «ХПІ», 2024. – С. 193-194.

9. Gruzdo I., Kyrychenko I., Tereshchenko G., Shanidze O. Analysis of Models Usability Methods Used on Design Stage to Increase Site Optimization // CEUR Workshop Proceedings. – 2023. – № 3403. – С. 387-409.

16. Gruzdo I., Kyrychenko I., Tereshchenko G., Shanidze N. Metrics Applicable for Evaluating Software at the Design Stage // 5th International Conference on Computational Linguistics and Intelligent Systems (COLINS-2021), April 22–23, 2021, Kharkiv, Ukraine. CEUR Workshop Proceedings. – 2021. – Т. 2870, Вип. I. – С. 916-936.

20. Shirokopetleva M., Shevchenko O., Pervieieva D., Savulioniene L., Sakalys P. A Comparative Analysis of Native iOS Frameworks for Developing Graphics Scenes in Augmented Reality Applications // Environment. Technology. Resources. Proceedings of the 16th International Scientific and Practical Conference – Rezekne, Latvia, 2025. – Vol. II. – С. 321–327. URL: https://doi.org/10.17770/etr2025vol2.8610