Міністерство освіти і науки України НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
«КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

Створення інтерактивних інструкцій для пристроїв за допомогою доповненої реальності

Текстова частина до курсової роботи за спеціальністю «Інженерія програмного забезпечення» - 121

Керівник курсової роботи

к.т.н. ст. викл. Франків О. А.

_____

(Підпис)

“___” _____ 2022 року

Виконала студентка ІПЗ-4

Сорокопуд Ю.М.

“___” _____ 2022 року

Київ 2022

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ

Зав.кафедри інформатики,

доцент, к.ф-м.н.

_____ С. С. Гороховський (підпис)

„____”_____2022 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студентки Сорокопуд Юлії Миклаївни факультету інформатики 4-го курсу

ТЕМА: Створення інтерактивних інструкцій для пристроїв за допомогою доповненої реальності

Зміст ТЧ до курсової роботи:

Індивідуальне завдання

Вступ

Частина 1: Аналіз предметної області. Постановка завдання курсової роботи

Частина 2: Теоретичні відомості

Частина 3: Опис реалізації програмного продукту

Висновки

Список літератури

Дата видачі „___” _____ 2022 р. Франків О.А.

_____(підпис)

Завдання отримала _____

(підпис)

Календарний план виконання роботи:

Тема: **Розробка мобільного додатку-помічника по приготуванню їжі з використанням технологій комп‘ютерного бачення**

No	Назва етапу	Термін виконання	Примітка
1	Отримання завдання на курсову роботу	20.10.2021	
2	Пошук тематичної літератури	28.10.2021	
3	Огляд літератури за темою роботи	15.11.2021	
4	Проведення дослідження	17.02.2022	
5	Аналіз отриманих результатів	10.04.2022	
6	Проектування архітектури додатку	18.04.2022	
7	Створення інтерфейсу додатку	19.04.2022	
8	Створення функціоналу доповненої реальності	11.05.2022	
9	Написання текстової частити курсової роботи	17.05.2022	
10	Здача курсової роботи	20.05.2022	

Студентка Сорокопуд Ю.М.

Керівник Франків О. А.

“____” _____

ЗМІСТ

ВСТУП *6*

Структура роботи 7

Постановка задачі 7

Розділ 1. Аналіз предметної області та постановка завдання *8*

1.1 Аналіз сучасного питання 8

1.2. Аналіз існуючих додатків інтерактивних інструкцій з використання доповненої реальності 8

Висновки до розділу 1 17

Розділ 2. Теоретичні відомості *18*

2.1 Використання фреймворку ARKit в iOS застосунках 18

2.2 Використання фреймворку RealityKit в iOS застосунках 23

Висновки до розділу 2 26

Розділ 3. Опис реалізації продукту *27*

3.1. Аналіз технічного завдання 27

3.2 Огляд засобів розробки 27

3.3. Опис та обґрунтування архітектури проекту 28

3.4 Опис розробки застосунку 30

3.5 Принцип роботи готового застосунку 33

Висновки по роботі та аналіз можливостей для подальшого розвитку застосунку *38*

Список джерел *40*

Анотація

Робота була присвячена розробці iOS додатку з використанням технологій доповненої реальності для створення додатку інтерактивних інструкцій Для розробки додатку були використані мова Swift, фреймворк ARKit, RealityKit.

Ключові слова: доповнена реальність, AR, iOS

ВСТУП

Швидкість розвитку технологій за останні десятилітня важко переоцінити. Щороку все більше людей купують собі смартфони, так з 2017 року до 2022 роки, кількість власників смартфонів збільшилась майже на 50%.[1] І кількість проведеного часу за в інтернеті в середньому становить 6:40-7 годин. Якщо ми так багато часу проводимо в Інтернеті, чи можна сказати що ми живемо на два світи – реальний та віртуальний? Мабуть, так.

Але чи варте створювати окремий світ, якщо можна поєднати два та отримати з обох найкраще? На сьогодні до 26% смартфонів підтримують технології ARKit та ARCore. Ще в 2020 83.1M американців використовували доповнену реальність щомісячно. Всього в 2021 році уже було 192 мільйони активних користувачів ARKit та 182 мільйони ARCore, порівняно з 147 та 122 мільйонами відповідно в 2020. Лише за один рік додались 105 мільйонів унікальних користувачів. [2]

.Технології доповненої та віртуальної реальності розвиваються неймовірно швидко. І найкраще, що завдяки ним вирішується дуже багато «больових точнок» користувачів. У кожного було таке, коли купити щось додому з інтер’єру, але не впевнений чи підійде по дизайну чи розміру? IKEA скористалась цим і випустила додаток, який уже завантажити близько 31,3 разів.

Чи часто було таке, що інструкція яка іде разом з продуктом особливо нічим не допомагає? Деталі виглядають однаково і не зрозуміло звідки починати. Рішенням був би додаток інтерактивних інструкцій, який би крок за кроком описав би інструкцію при цьому точно показуючи, що та де робити.

Виходячи з даної потреби користувачів та актуальності і можливості впроваджувати технології доповненої реальності в застосунки, за мету буо поставлене створення інтерактивних інструкцій для пристроїв за допомогою доповненої реальності

Структура роботи

Робота складається з вступу та трьох основних розділів, висновків, списку джерел та додатків.

У першому розділі описаний аналіз предметної області.

У другому розділі описані основні технології, які використовувались для розробки додатку, теоретичні відомості про них.

У третьому розділі описані етапи розробки та принципи роботи готового застосунку.

Постановка задачі

1. Проаналізувати додатків-аналогів даної предметної області
2. Проаналізувати принципи роботи доповненої реальності.
3. Дослідити технології графічного відображення віртуального вмісту доповненої реальності.
4. Використовуючи мову Swift та досліджені технології, створити iOS-додаток інтерактивних інструкцій.

Розділ 1. Аналіз предметної області та постановка завдання

1.1 Аналіз сучасного питання

Доповнена реальність має величезний потенціал практично в кожній сфері: навчанні, медицині, бізнесі, будівництві і т,д.. Активно можна використовувати її ще в рутинних справах. Пізніше будуть розглянуті додатки-аналоги застосунку який ми будемо розробляти. Ними виявилися великі промислові компанії, де інструкції уже зазвичай наперед підготовлені.

Це дуже зручно для тих конкретних цілей, однак недостаток їх в тому, що їх неможливо перевикористовувати. Тому було вирішено написати власний застосунок, завдяки якому користувач зможе створити інструкції в доповненій реальності.

1.2. Аналіз існуючих додатків інтерактивних інструкцій з використання доповненої реальності

Перш ніж приступати до розробки самого додатку, був проведений аналіз конкурентів, щоб проаналізувати переваги та недоліки їх застосунків.

Рисунок 1.1 AR Instructor від ARVisio

AR Instructor від ARVisio

AR Instructor використовує технології доповненої та змішаної реальності.. AR Instructor надає інструкції на робочому місці, використовуючи попередньо підготовлені покрокові інструкції з доповненої реальності. Користувачеві надаються візуальні підказки, щоб направляти користувача під час виконання заданого набору завдань. Кожен крок керованого робочого процесу можна проілюструвати за допомогою відео, документів, зображень і 3D-моделей, накладених на об’єкти в полі зору користувача.

Великий набір інструкцій різного типу покращує досвід користувача, це є перевагою. Однак неможливість керувати або додавати зміни інструкціями є суттєвим мінусом.

Рисунок 1.2. CareAR Assist

CareAR Assist

CareAR Assist дозволяє сервісним командам у будь-якому місці миттєво надавати віддалену візуальну підтримку AR для своїх клієнтів та співробітників. Надає доступ до віддалених експертів.

Основною перевагою є спільна робота двох людей в мережі.

1.3 Основні поняття технологій доповненої реальності

Концепція доповненої реальності

Доповнена реальність (з англ. AR - Augmented reality) як концепція передбачає реальність із додаванням трохи віртуальної частини. На відміну від віртуальної реальності, вона не створює повністю новий цифровий світ навколо нас, а вдосконалює справжній. Вона може покращити реальні речі за допомогою цифрової інформації, додати комп'ютерні об'єкти або зробити звичайні речі інтерактивними та маніпулятивними. З AR цифрова інформація стає частиною реального світу, принаймні, у сприйнятті користувача.

Найбільш загальноприйняте визначення AR було запропоноване Azuma в його оглядовій статті 1997 року. Відповідно до Azuma [1997], будь-яка доповнена реальність повинна мати такі три характеристики[3]:

1. Вона поєднує в собі два світи: віртуальний та реальний, де реальний - основне місце дії.
2. Створений "світ" має бути інтерактивним та оновлюватися в реальному часі
3. Віртуальна інформація має зберігатися в трьох вимірному просторі

Тобто у нас не має зобов'язання щодо дисплею чи будь-якого іншого візуального засобу. Чого ми справді маємо дотримуватися - інтерактивності та відповідності нашої віртуальної сцени до реальної світової картинки. Людина з комп'ютером мають існувати у тісно пов'язаному циклі. Користувач змінює позицію в сцені та керує віртуальними об'єктами. А система відслідковує свою позицію в реальному світі з віртуальним вмістом, а потім представляє користувачеві візуалізацію.

Тож для створення системи доповненого бачення нам потрібні три складові:

- компонент відстеження - дані з реального світу
- компонент реєстрації - дані з віртуального світу
- компонент візуалізації – графічне представлення

Всі дані для компонентів відстеження та реєстрації зареєстровані в одній системі координат

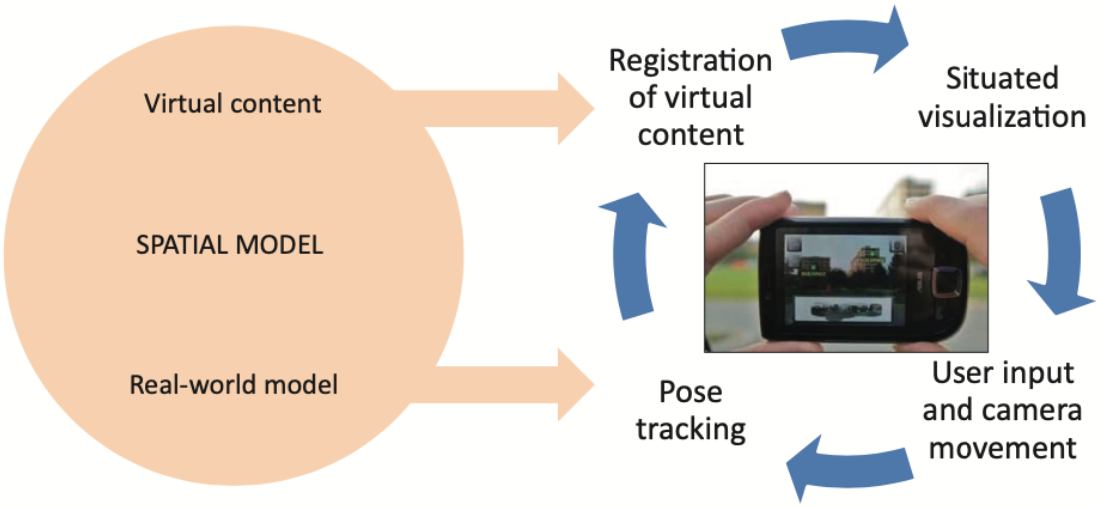


Рисунок 1.3. .Цикл зворотного зв'язку між користувачем і комп'ютерною системою. [4]

AR використовує цикл зворотного зв'язку між користувачем і комп'ютерною системою. Користувач спостерігає за дисплеєм AR і керує точкою огляду. Система відстежує точку зору користувача, реєструє позу в реальному світі разом із віртуальним вмістом і представляє локалізовані візуалізації.

Основні поняття доповненої реальності

Доповнена реальність успадковує багато концепцій і термінології від комп'ютерної графіки. Серед таких:[5]

- Полігональна сітка (з англ. Polygon mesh) — це набір вершин, ребер, та граней, що описують форму багатогранного об'єкта в тривимірній графіці. Кількість граней всередині сітки відома як кількість полігонів.
- Модель — одна полігональна сітка або набір сіток, вона включає дані про текстуру, освітлення, анімацію і т.д.. Роздільна здатність моделі залежить від кількісті точок у сітці, наприклад, «низькополігональна» модель має менше точок, які, у свою чергу, утворюють менше граней.
- Сцена — колекція 3D-моделей з атрибутами, ієрархія яких між собою представлена в графі сцени .
- Рендерер (з англ. Renderer) — програма, яка створює двовимірне візуальне зображення віртуальної сцени; У доповненій реальності рендерер використовує камеру та датчики руху, щоб захопити точку зору користувачів і точно намалювати віртуальну сцену.
- Трекінг (з англ. Tracking) — для виявлення точок для відстеження у вихідному відео (це можуть бути риси обличчя, об’єкт, 2D-зображення або сам світ), а потім на основі них точно відмальовується віртуальний вміст сцени з потрібним масштабом і поворотом.

Також варто згадати термінологію доповненої реальності:

Маркер — об’єкт реального світу, який використовується для прив’язки об’єків доповненої реальності.

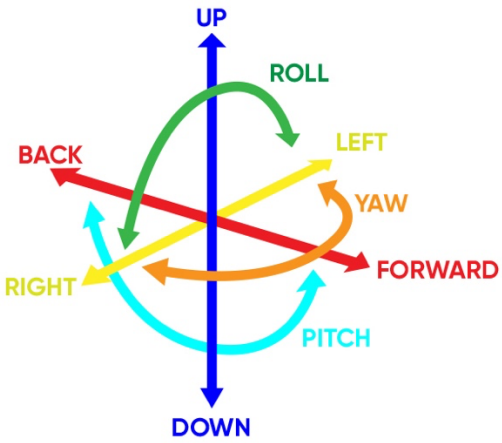


Рисунок 1.4. 6DoF

- DoF [6]— ступені свободи (з англ. Degrees of Freedom) — це система, яка використовується для опису здатності об'єкта переміщатися у своєму просторі. Існують різні рівні DoF:
    - 3DoF — об'єкт з 3DoF (трьома ступенями свободи) може обертатися лише на трьох осях (на рисунку roll, yaw, pitch); змінюється лише орієнтація об’єкта,сам він залишається нерухомим.
    - 6DoF — об'єкт з 6DoF (шістьма ступенями свободи) може рухатися вперед і назад; догори й донизу; і ліворуч і праворуч на додаток можливостей 3DoF (усі напрямки на рисунку) Ці додаткові можливості дозволяють краще досліджувати свій простір.
- Оклюзія — стан або умова, коли об’єкт прихований іншим, оскільки він знаходиться за цим об’єктом у тривимірному просторі.

Типи доповненої реальності

Існують такі типи доповненої реальності: [7]

- AR на основі маркерів (з англ. AR based on markers)
- AR без маркерів (з англ. AR without markers)
    - На основі локації (з англ. Location-based AR)
    - На основі проекції (з англ. Projection-based AR.)
    - Накладання (з англ. Overlay AR.)

AR на основі маркерів

В цьому випадку для відображення об’єктів на сцені потрібно спочатку розпізнати зображення чи об’єкт (маркер) і залежно від його позиції, повороту, розміру накладе на нього 3D цифровий вміст. Надалі саме такий тип доповненої реальності використовуватиметься в практичній частині.

AR без маркерів

На відміну від доповненої реальності на основі маркерів, така дозволяє розташовувати 3D вміст отримавши інформацію з GPS, компасу, гіроскопу чи акселерометру. Таким чином, такий підхід працює з цифровими даними, отриманими цими датчиками, здатними записувати фізичний простір в режимі реального часу. Насамперед, аналіз без маркерів використовує SLAM.

Для доповненої реальності пристрій має знати більше: його 3D положення у світі. Він обчислює це через просторове співвідношення між собою та кількома ключовими точками, знайдених завдяки камер і датчиків. Цей процес називається «одночасна локалізація і картографування» (з англ. Simultaneous Localization and Mapping – скорочено SLAM).

SLAM[8] без попередніх знань про навколишнє фізичне середовище дозволяє програмам виявити об’єкти, точки, перепони і зрозуміти, як правильно розмістити віртуальні об’єкти на сцені.

AR на основі локації

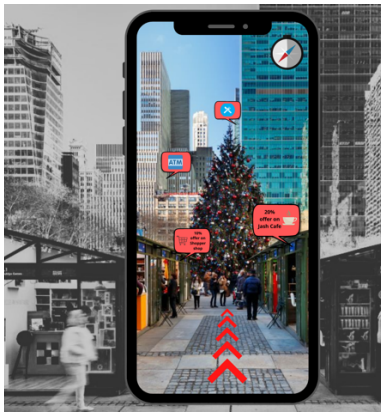
Ця технологія використовує місцезнаходження, щоб розташувати віртуальний об’єкт у бажаному місці. Найвідомішим прикладом цього типу доповненої реальності є гра Pokémon GO. AR на основі локації так може використовуватися в туристичних додатках для позначення визначних пам’яток

AR на основі проекції

Доповнена реальність створюється шляхом відео-проектування на реальну площину. Таким чином завдяки штучному світлу створюється ефект глибини, орієнтації та положення об’єкта.

AR накладання

Ідея доповненої реальності з накладанням полягає в заміні оригінального вигляду об’єкту оновленим віртуальним. Таку AR дуже часто можна зустріти в музеях, коли при на веденні на картину її вміст анімується.



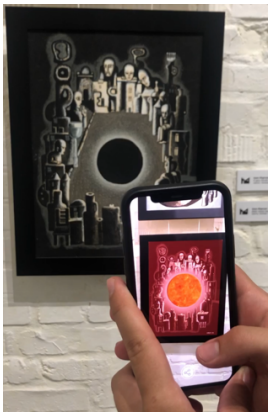


Рисунок 1.5. Приклад навігаційного додатку з AR на основі локації

Рисунок 1.6. AR додаток з накладанням в Музеї сучасного мистецтва Корсаків

Рисунок 1.7. Використання доповненої реальності на основі проекції для навчання

Огляд технологій для реалізації додатку

Для початку роботи над будь-яким додатком доповненої реальності, треба базове розуміння геометрії та 3D-графіки. Далі варто почати з вибору платформи орієнтуючись на поставлені задачі, бюджет, розуміння інструментів розробки, потреби цільової аудиторії.

Є багато різних платформ для написання програм доповненої реальності. Одні дозволяють розробляти застосунок для ПК, інші для мобільних пристроїв на різних операційних системах. WebVR і WebAR також стають все більш популярними, їх можна розробляти за допомогою A-Frame і Javascript.

Серед найпопулярніших платформ для розробки AR є:

Unity – одна з найвідоміших VR-платформ для розробки ігор. Однак вона також дуже ефективна для створення віртуальної реальності. Для розробки використовується мова C#.

Unreal Engine – використовуючи C++, фреймворк Unreal Engine AR надає величезні можливості для створення додатків доповненої реальності з Unreal Engine для платформ iOS і Android.

Vuforia — це ще одна потужна платформа AR, спрямована на розробку додатків AR. Vuforia надає API (application programming interface – з англ. «прикладний програмний інтерфейс») на мовах C++, Java, ObjC та .NET через розширення Unity. Таким чином, SDK підтримує розробку для iOS, Android і UWP, а також дозволяє розробляти кросплатформенні додатки AR в Unity

Google ARCore – надає різноманітні інструменти для розуміння об'єктів у реальному світі. Ці інструменти включають розуміння навколишнього середовища, що дозволяє пристроям виявляти горизонтальні та вертикальні поверхні та площини. Працює з Unity3D і Unreal Engine, а також з пристроями Android, використовуючи мову програмування Java.

Apple ARKit [9] – був представлений у червні на WWDC 2017. Платформу для створення додатків для iOS. ARKit поєднує в собі відстеження переміщення пристрою, зйомку сцени камерою, розширену обробку сцени та зручності відображення, щоб спростити завдання створення доповненої реальності. Ви можете створити багато видів AR за допомогою цих технологій, використовуючи передню або задню камеру пристрою iOS.

Тож, для операційних систем iOS та Android є нативні інструменти ARKit та ARCore відповідно. Випущені лише в 2017 та 2018 роках, фреймворки розвинулися та забезпечують майже всі можливості для розробки AR додатків на мобільні пристрої. Платформи Apple і Google конкурують один проти одного. Однак поки ARKit має значну перевагу. Датчик LiDAR є однією з найсучасніших технологій визначення глибини на мобільному пристрої, доступних на даний момент.

Для виконання поставленої задачі було вирішено розробляти нативний для iOS додаток, оскільки для цього є необхідне розуміння фреймворку для розробки - ARKit та наявні девайси Apple.

Висновки до розділу 1

В першому розділі були наведені знайдені та проаналізовані аналоги-конкуренти додатку. Також були розглянути базові принципи доповненої реальності, визначені типи доповненої реальності та обрано який з них використовуватиметься в подальшій розробці додатку. Далі були знайдені та розглянуті найпопулярніші платформи для розробки застосунків з використанням доповненої реальності після чого обрано один з них ARKit для реалізації проекту.

Розділ 2. Теоретичні відомості

2.1 Використання фреймворку ARKit в iOS застосунках

В 2017 році на щорічній конференції Apple WWDC був представлений новий фреймворк – ARKit. ARKit поєднує в собі всі необхідні технології для простого створення та ефективної роботи доповненої реальності.

Робота фреймворку складається з трьох тісно пов’язаних між собою етапів: відслідковування (з англ. tracking), розуміння сцени та рендеринг. [10]

Відслідковування

ARKit – найголовнішою властивістю роботи фреймворку є *ARSession*. Тому він є основаним на сесії (з англ. session based). Якщо використовується власний власний засіб візуалізації, для роботи потрібно буде самостійно створити сесію – ARSession. Якщо ж використовується один із стандартних засобів візуалізації (наприклад, ARView, ARSCNView або ARSKView), засіб візуалізації сам створює для вас об’єкт сесії. Основа взаємодія з нею відбувається в інструменті візуалізації програми.

ARSession[11] – це об’єкт, який контролює всі процеси створення нашого додатка доповненої реальності. Для нього потрібно задати конфігурацію – ARConfiguration. Сесія може запускати лише одну конфігурацію за раз, тому на цьому етапі треба обрати ту, яка буде використовуватись в подальшій роботі. Apple надає нам довгий список можливих конфігурацій, серед них є:

- AROrientationTrackingConfiguration – забезпечує три ступені свободи відстеження, що є лише орієнтацією вашого пристрою.
- ARWorldTrackingConfiguration – забезпечує шість ступенів свободи відстеження. Таким чином, отримується не лише орієнтацію вашого пристрою, а відстежується як його положення, так і орієнтація щодо будь-яких поверхонь, людей або заданих зображень та об’єктів.

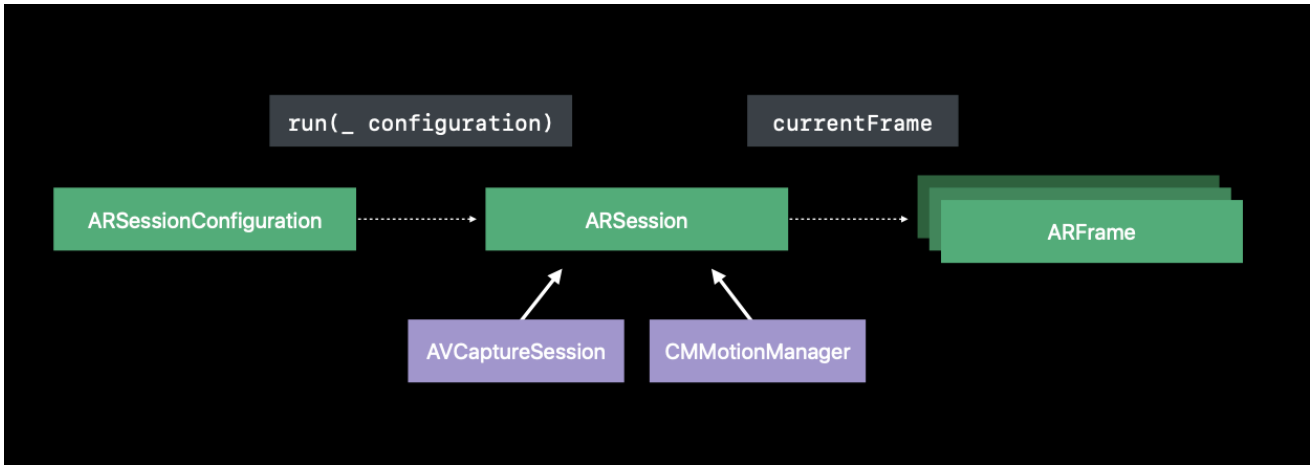
- ARImageTrackingConfiguration – відстежуються лише задані зображення – trackingImages.

Після створення екземпляру конфігурації необхідно запустити нашу сесію. Для цього, передати конфігурацію в метод run, що викликається на ARSession.

```
let configuration = ARWorldTrackingConfiguration()
self.session.run(configuration)
```

І з цього моменту починається трекінг позиції та обертання девайсу в середовищі.

Під капотом ARSession створює AVCaptureSession та CMMotionManager , щоб отримати дані зображення і руху, які будуть використовуватися для відстеження. Для розпізнавання та обробки зображень, об'єктів, людей також використовується фреймворк машинного навчання Apple Core ML.



Русунок 2.1. Схема роботи ARSession

Результатом відпрацювання ARSession є об'єкти ARFrame.

Об'єкт *ARFrame* надає інформацію про відстеження або орієнтацію пристрою користувача, а також місцезнаходження та навіть стан відстеження. Доступитися до нього можна напряму витягнувши з нашого екземпляру ARSession. Або

інплементувати ARSessionDelegate та слідкувати за оновлюваним останнім фреймом в методі func session(_ session: ARSession, didUpdate frame: ARFrame).

У випадках якщо сцену уже не видно, можна поставити відслідковування на паузу викликавши pause() на сесію. Щоб відновити відстеження після паузи, можна просто знову викликати run() зі збереженою конфігурацією вашого сеансу.

Також можливо переключатися між різними конфігураціями для цього треба просто викликати знову метод run(), задавши потрібну нам конфігурацію. Завдяки цьому ж методу можна скинути всі дані про позицію, наявні відстежені якорі, для цього передаємо необхідну опцію в RunOptions.

Кожен екземпляр ARFrame містить в собі ARCamera – що представляє віртуальну камеру. ARCamera містить в собі transform.

Transform — це матриця або float 4x4, яка зберігає всі дані про перетворення (зміна орієнтації, позиції) фізичного пристрою відносно початкової точки сесії.

Окрім цього, у ARCamera властивість стану відстеження статусу трекінгу (з англ. TrackingState) з трьома можливими значеннями: недоступний, нормальний та обмежений (з англ. notAvailable, limited та limited відповідно)

Коли сесія тільки починається TrackingState завжди недоступний, це означає що після початку у сесії ще недостатньо даних для обробки положення пристрою. Як тільки цих даних стає достатньо щоб обрахувати знаходження гаджету - стан TrackingState зміниться з «недоступний» на «нормальний». Пізніше стан може змінитись на обмежений, надавши при цьому причину зміни стану.

Розуміння сцени

Отримавши всю необхідну інформацію з ARFrame завдяки трекінгу, ми можемо перейти до наступного етапу - розуміння сцени.

Мета розуміння сцени — дізнатися більше про наше середовище, щоб помістити віртуальні об’єкти в це середовище. Саме на етапі розуміння сцени ми визначаємо середовище на якість його освітленості, щоб могти реалістично розмістити там віртуальний об’єкт.

Тут з’являється нова сутність – ARAnchor [12]. Це відносне або реальне положення та орієнтація в просторі. Воно потрібне для коректного представлення віртуального вмісту в сцені і ми можемо довільно додати та видаляти ARAnchor з нашої сцени. Якорі прив’язуються до реального фізичного середовища. Ще ARAnchors додаються до сесії автоматично, якщо ми використовуємо розпізнавання наприклад площини.

Одним з підкласів ARAnchor є ARImageAnchor. Такий Anchor додається до сцени ARKit шукає в реальному світі одне з зображеннь з ARReferenceImage які задаються при налаштуванні конфігурації, та після розпізнання зображення автоматично додає до списку якорів сесії ARImageAnchor. Після цього на створений anchor можна буде додавати віртуальні моделі. До того ж, щоб ці віртуальні моделі правильно розмістити в просторі і масштабувати, при створенні ARReferenceImage для кожної фотографії має задаватися її фізичний розмір у метричній системі. Завдяки цьому фреймворк може правильно обчислити відстані та розміри у нашій сцені.

Для чіткого позиціювання ARAnchor на сцені ми використовуємо Raycasting [13]. Тепер завдяки інноваційному датчику LiDAR, який є на iPad Pro, Phone 12 Pro, 12 Pro Max, 13 Pro, та 13 Pro Max, користувач більше не повинен рухати його девайсом, щоб «допомогти» програмі вивчити навколишнє середовище. Відскановане з інформацією про глибину, ARKit конвертує в полігонну сітку, яка дозволяє набагато точніше розміщувати віртуальні об’єкти на різноманітних поверхнях з урахуванням їх кривизни та кута.

Також варто згадати про оклюзії людей та об’єктів в ARKit (з англ. occlusion). Ефект допомагає розпізнати фігуру за допомогою оклюзії на основі глибини та гарно

відображати 3D контент в віртуальному світі, заховати за людиною чи об’єктом, якщо треба. Цей ефект значно кращий на нових пристроях, оснащених сканером LiDAR.

Аналіз освітлення навколишнього середовища увімкнений за замовчуванням. Інтенсивність нормально освітленого зображення становить 1000 люмен. Значення інтенсивності світлішого і темнішого зображень збільшуються та зменшуються відповідно.

Якщо до сцени було додано віртуальний вміст, то він підлаштується під навколишнє середовище. Наприклад, якщо зробити світло у кімнаті тьмянішим, ваш віртуальний контент також тьмяніє.

Рендеринг

І останнім після трекінгу та розуміння сцени залишився рендеринг. Однак за нього, як і за завантаження 3D-моделей фреймворк ARKit не відповідає. Тому Apple пропонують декілька інших нативних фреймворків для рендерингу і навіть відносно просту інтеграцію з популярними ігровими рушіями, зокрема:

- *SpriteKit*: високорівневий фреймворк для відображення двовимірних графічних елементів: фігур, тексту, анімацій та відео. Отже вся графіка реалізована завдяки цьому фреймворку завжди буде розвернута до камери. Для поєднання SpriteKit та ARKit, ми використовуємо клас ARSKView.
    - *SceneKit*: високорівневий фреймворк для створення тривимірних сцен і спеціальних ефектів з геометрією, матеріалами, світлом, системами частинок і камерами. Використовуємо клас ARSCNView для поєднання SceneKit та ARKit
    - *RealityKit*: високорівневий 3D-графічий фремворк, створений з нуля на основі Metal спеціально для рендерингу доповненої реальності. Він пропонує роботу з 3D об’єктами, матеріалами, анімацією, звуком. Ми використовуємо клас ARView для поєднання RealityKit та ARKit

- *Metal*: графічний API найнижчого рівня від Apple, що забезпечує розробникам найшвидший і найпряміший доступ до обробки графіки. Для створення власної візуалізації.
- *Unity & Unreal*: Apple із спільнотою розробників ігор створення ряд плагінів для популярних ігрових рушіїв, таких як Unity та Unreal

2.2 Використання фреймворку RealityKit в iOS застосунках

З самого початку для графіки в iOS розробці для використовувася SceneKit. Однак теперішні вимоги до AR вимагають нових рішень, які SceneKit на жаль не завжди може реалізувати. Просто додати вміст до доповненої реальності та взаємодіяти з ним сьогодні недостатньо. Зараз кожне віртуальне представлення повинне виглядати максимально наближено до рельного. Користувач очікує дуже якісний рендеринг з точним позиціюванням, моделями з реалістичним освітленням, тінями, ефектами та просторовим звуком. RealityKit [14]— це високорівневий фреймворк рендерингу з дуже простим API, розроблений з нуля одразу орієнтуючись на доповнену реальність. Разом з ним був представлений Reality Composer, інструмент для Mac і iOS, який дозволяє дуже просто створювати контент для AR.

RealityKit щільно інтегований з Reality Composer. RealityKit має можливість працювати з двома файловими форматами Reality Composer [15]– .rcproject та .reality та крім цього з нативним форматом– .usdz. Але використання Reality Files дасть вам набагато швидший час завантаження.

RealityKit має вбудовану підтримку для роботи в мережі, розроблену для спільного досвіду користувачів. Він пропонує автоматичну синхронізацію об’єктів між кількома девайсами. Також завдяки просторовому аудіо можна зробити досвід використання доповненої реальності ще більш "реальним". Можна приєднувати звукові ефекти до 3D-об'єктів, а потім відстежувати ці звуки, щоб вони звучали реалістично на основі їхнього положення в реальному світі. В RealityKit можна

налаштувати фізичні властивості для віртуальних об’єктів, наприклад масу, щоб точно симулювати їх поведінку на реакцію зіткнення з іншими об’єктами

Отже, використовуючи RealityKit, потрібно розуміти що таке та за що відповідають *ARView, Scene, Entity, Component.*

ARView,

Основне, що треба розуміти, ARView - це клас, що поєднує наш графічний RealityKit та ARKit та відображає контент доповненої реальності.

Scene (Сцена)

Це контейнер який зберігає всі сутності для рендерингу. Вона автоматично створюється в ARView в одному екземплярі. До неї можна додати декілька якірних сутностей, прив’язуючи їх до об’єктів реального світу.

Entity (Сутність)

До сцени додаються сутності. По суті весь представлений віртуальний вміст є сутностями. До них приєднуються компоненти (з англ. Component), щоб визначити зовнішній вигляд і характеристики поведінки. Кожна сутність за замовчуванням має компоненти Transform та Synchronization, до відповідають за перетворення (масштаб, поворот і зсув) та за її синхронізацію в мережі відповідн

RealityKit має багато попередньо визначених компонентів, які ви можна використовувати. Позначити що сутність може прив’язуватися до чогось (AnchorComponent), зіткнутися з іншими сутностями(CollisionComponent), мати якийсь окремо визначений зовнішній вигляд (ModelComponent) – це все визначається компонентами.

Також кожну сутність можна передавати іншій сутності як нащадок. Таким чином поведіка доданої поверх сутністі буде залежити від батьківської.

AnchorEntity (Якірна сутність)

За допомогою AnchorEntity ви вказуєте, до чого він має бути прив'язаний у реальному світі. І коли цей об’єкт рухається у вашому середовищі, AnchorEntity залишатиметься приєднаним до нього. Зазвичай саме AnchorEntity буде основою ієрархії сутностей, на його основі створюється додатковий вміст. AnchorEntity підтримує всі типи якорів, доступні в ARKit. Це дозволяє швидко перенести свій вміст у реальний світ прив’язавшись до будь-чого.

Важливо, що якщо ARKit ще не виявив прив’язку, яка відповідає цій даній сутності, вона залишиться неактивною.А потім, як тільки ми виявимо прив’язку, яка йому відповідає, як AnchorEntity, так і всі його children стануть активними.

ModelEntity

ModelEntity – саме вона визначає те, що ми бачимо. ModelEntity є наповнювачем. З точки зору компонента сутності, ModelEntity — це сутність із компонентом моделі, фізичним компонентом та компонентом зіткнення. Такі об’єкти можна створювати динамічно в коді або завантажувати їх безпосередньо з USDZ або Reality File. Після її завантаження варто просто додати її до AnchorEntity і вона з’явиться перед юзером.

При ініціалізації ModelEntity, ми передаємо *полігонну сітку* (mesh), яка забезпечує геометричне представлення моделі та матеріали, які описують зовнішній виглад.

Ресурси Mesh можуть бути згенеровані безпосередньо як примітив чи завантажена я з USDZ або Reality File модель. Говорячи про примітиви, мається на увазі поля, сфери, площини та тексти, які підтримують усі шрифти на нашій платформі.

Матеріали забезпечують зовнішній вигляд об’єкта і те, як він взаємодіє зі світлом навколо нього.

Матеріали можуть бути або попередньо визначені створені власноруч та передані в ModelEntity. Наприклад є такі матеріали:

SimpleMaterial задається за допомогою трьох основних властивостей: основного кольору, шорсткості та металевого кольору. Колір задає колір. А параметр шорсткості впливає на шорсткість об’єкта та на те, як світло відбивається від поверхні об’єкта. А металевий параметр фактично моделює, наскільки провідним є цей матеріал, і також впливає на те, як світло взаємодіє з цією поверхнею.

UnlitMaterial – варто звернути і на нього увагу. Матеріал, який має лише колір і його зображення ніяк не залежить він зовнішнього світу. Добре підходить до пласких об’єктів в доповненій реальності, як текст. Або об’єктів які в фізичному світі не залежить від світа, наприклад екран комп’ютера

OcclusionMaterial – маскувати, ховати та трошки приховувати це все можна зробити з об’єктом закдяки цьому матеріалу.

Анімація

Анімація моделей зазвичай йде в самому файлі моделі, однак з коду можна додати анімацію перетворень. Задавши час, амплітуду на запустивши playAnimation на потрібний об’єкт.

Висновки до розділу 2

В другому розділі був розглянутий фреймворк для роботи з доповненою реальністю на iOS – ARKit, досліджено принципи його роботи . Було виділені основні фреймворки для рендерингу віртуального вмісту, які використовуються разом з ARKit, оскільки останній сам не підтримує цієї функції. Після чого було обрано та розглянуто фреймворк для рендерингу RealityKit .

Розділ 3. Опис реалізації продукту

3.1. Аналіз технічного завдання

Технічним завданням є розробка iOS додатку з технологією доповненої реальності для створення інтерактивних інструкці.

Необхідно розробити додаток, завдяки якому користувач створити інтерактивну інструкцію, яка з’являється на його екрані при наведенні на обране зображення у відносному до нього положенні. У споживача також має бути можливість створити власну інструкцію, задавши перед цим зображення до якого ця інструкція, або набір інструкцій будуть прив’язані.

Також потрібно створити інтуїтивно-зрозумілий користувацький інтерфейс, для полегшення взаємодії користувача з додатком.

3.2 Огляд засобів розробки

Як середу розробки було обрано Xcode – IDE розроблену Apple. Xcode має весь необхідний набір інструментів, для проектування, написання, тестування та створення інтерфейсу додатків для платформ Apple.

Xcode надає дві можливості мануального тестування: через симулятор та просто підключивши фізичний девайс до комп’ютера через кабель. Симулятори емулюють роботу пристроїв iPhone, iPad, Apple Watch та Apple TV. Якщо потрібно протестувати додаток на різних операційних системах, можна додатково скачати ситулятори на потрібній ОС.

Але оскільки симулятори не можуть емулювати роботу доповненої реальності, під час розробки додатку тестування відбувалося цілком на девайсі iPhone 12 Pro Max.

ARKit — це нативний фреймворк від Apple для розробки доповненої реальності для мобільних пристроїв iOS.

RealityKit — це нативний фреймворк від Apple для рендерингу графічних елементів доповненої реальності.

SwiftUI [16] — це новий фреймворк від Apple для створення інтерфейсів користувача для iOS, tvOS, macOS і watchOS, представлений Apple у 2019 році.

Firebase [17] — це мобільний бекенд як сервіс (з англ. Backend-as-a-Service - Baas), який надає потужні функції для створення мобільних додатків. Firebase має три основні служби: база даних реального часу, аутентифікація користувача та хостинг.

Swift Package Manager[18] – це нативний менеджер залежностей Apple, він вбудований в Xcode.

3.3. Опис та обґрунтування архітектури проекту

MVVM[19] – Model - View – View Model був обраний як шаблон проектування нашого застосунку. Шаблони в програмуванні потрібні для того, щоб розподіляти обов’язки між класами, таким чином код набагато легше читається і розуміютьсяі. В MVVM собою представляє три сутності:

Model – в ній представлені моделі даних, вона повідомляє про реальний стан даних.

View Model – зв’язує дані Model та їх відображення у View та відслідковує поточний стан даних Model. View Model не має доступу до View.

View – відповідає за макет, структуру, зовнішні вигляд інтерфейсу користувача та обробку взаємодії користувача з додатком, повідомляє View Model про зміни.

При цьому зв’язок між рівнями рівнями проектування в MVVM будується «зв’язуванням даних» (binding).

Binding – забезпечує зв’язок між властивістю, що зберігає дані, та відображенням (View), яке їх відображає. [17]

26

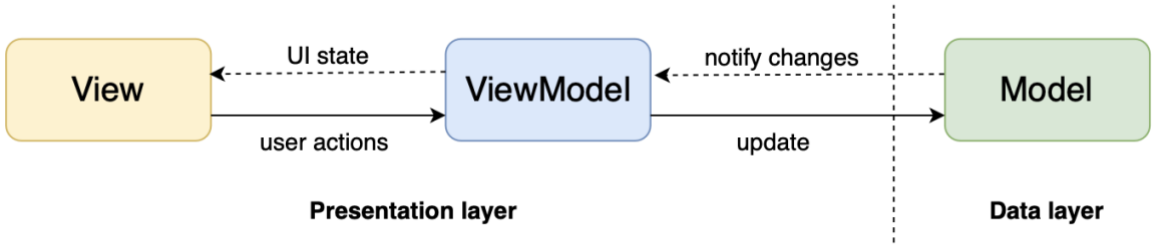


Рисунок 3.1. Схема шаблону проектування MVVM

З виходом SwiftUI, стало дуже зручним використовувати саме MVVM архітектуру, адже концепція «зв’язуванням даних» лежить в її основі.

SwiftUI використовує обгортки(wrapper) властивостей, такі як @State, @Binding, @ObservedObject, @ObservableObject, @StateObject та @Published. View залежить від стану властивостей і щоразу, як оновлюються дані, змінні у View також змінюються, а інтерфейс перебудовуються автоматично.

@State – завжди приватна властивіть, яка знаходить в середині нашого відображення. Коли ми передаємо її значення в наступну View як @Binding, вона є для нього джерелом істини.

@ObservedObject та @StateObject – практично мають однакові властивості, розниця в тому, що @StateObject – джерело інстини, а *@ObservedObject – об’єкт який спідкує за попереднім.*

@Published – приєднується до властивостей усередині ObservableObject і повідомляє відображення, що він повинен оновити змінені властивості.

3.4 Опис розробки застосунку

Основні принципу роботи додатку

Після запуску додатку, користувач може або відкрити раніше створену сцену з інструкціями, або створити нову. В першому одразу випадку відбудеться навігація до ARView. Зображення та збережні сутності разом з даними про прикріплені до них інструкції будуть підтягнути з Firebase.

Для нової сцени потрібно задати зображення та його фізичну ширину. Потім переходимо до ARViewта додаємо вибране зображення для відслідковування як ARReferenceImage в конфігурацію сесії.

Щоб додати нову інструкцію на сцену, треба спочатку ввести всі її дані а потім розмістити AnchorEntity на сцені, відносно до нашого зображення. Ця якірна сутність повинна зберігати своє положення відносно позиції зображення та оновлюватися відносно нього. Але додані інструкції мають зберігатися в самій сцені, щоб можна було додавати інструкції навіть коли зображення уже не в зоні бачення екрану.

Всі значення зберігатимуться на сервері Firebase.

Опис розробки застосунку

З самого початку після запуску програми ми робимо запит на Firebase і він нам повертає усі попередньо створені сцени з інструкціями. Класу ARScene Сцена являє собою клас, що імплементує протокол Identifiable.

Далі користувач може або створити нову сцену, або перейти до існуючих. Коли ми переходимо до відображення ARInstructionsSceneView, ми:

1. Створюємо ARView
2. Зберігаємо в локальній змінній нашу сцену

3. Назначаємо себе делегатом ARSessionDelegate

4. Додаємо зображення, яке має відстежуватися до множини Set<ARReferenceImage> передаючи окрім самої фотографії ще і її фізичну ширину (ширину користувач вводить під час створення сцени). Потім викликаємо метод run() на нашу ARSession

5. Робимо запит на сервер очікуючи інструкції, які можливо були раніше додані до сцени.

```
init(scene: ARScene) {
    self.arView = ARViewManager(frame: .zero)
    self.scene = scene
    super.init()
    arView.session.delegate = self
    addImageReference(scene: scene)
    getInstructions()
}
```

Рисунок 3.2 Ініціалізація на налаштування ARView

```
public func addReferenceImage(for image: UIImage, name: String, width: CGFloat) {
    guard let cgImage = image.cgImage else { return }
    let referenceImage = ARReferenceImage(cgImage, orientation: .up, physicalWidth: width)
    referenceImage.name = name

    self.newReferenceImages.removeAll()
    self.newReferenceImages.insert(referenceImage)

    configure()
}

func configure() {
    let configuration = ARWorldTrackingConfiguration()
    configuration.detectionImages = self.newReferenceImages
    self.session.run(configuration)
}
```

Рисунок 3.3 Додавання ARReferenceImage на налаштування конфігурації сесії

Після цього, якщо запит повернув якісь значення, інструкції локально зберігаються. Однак поки не буде розпізнано нашу ARReferenceImage, на екрані вони не з’являться.

Для того щоб додати нові сутності треба теж спочатку розпізнати ARImageAnchor, до якого вони мають прив'язатися. Навіть якщо спробувати це зробити – вони не з'являться на екрані.

```
/// Set rootAnchorEntity to ARImageAnchor
func session(_ session: ARSession, didAdd anchors: [ARAnchor]) {
    anchors.compactMap { $0 as? ARImageAnchor }.forEach {
        imageAnchor = $0
        arView.addRootAnchorEntity(for: $0) {
            getSavedEntitiesPositions()
        }
    }
}
```

Рисунок 3.4 розпізнання сесією ARImageAnchor та присвоєння до нього нашого rootAnchorEntity

Коли сесія розпізнає наш ARImageAnchor, прив'язуємо до нього нову AnchorEntity яка називається rootAnchorEnity, бо вона зберігає в собі всі інструкції як children. Після цього запитуємо всі раніше додані сутності на показуємо на екрані.

Щоб додати нову сутність (AnchorEntity) для інструкції до сцени, ми використовуємо метод addNewMarker(for instruction:completion:). В ньому ми створюємо ModelEntity – завдяки примітувам Mesh, які надає нам RealityKit, генеруємо маленьку прозору сферу. Для цього створюємо та присвоюємо сутності матеріал OcclusionMaterial(). Додаємо сферу до AnchorEntity(.camera) та змінюємо її позицію по осі z, щоб поставити цю сферу перед камерою. Додаємо нашу AnchorEntity як child до rootAnchorEntit, щоб зберегти його позицію відносно зображення.

```
    /// tracking image anchor transtorm and updating entities
    func session(_ session: ARSession, didUpdate anchors: [ARAnchor]) {
        anchors.compactMap { $0 as? ARImageAnchor }.forEach {
            arView.updateEntityForImageAnchor($0)
            self.imageAnchor = $0
        }
    }

    /// updating image anchor's and instructions' view positions
    func session(_ session: ARSession, didUpdate frame: ARFrame) {
        if let imageAnchor = imageAnchor {
            self.imageAnchorViewPosition = arView.imageAnchorPosition(of: imageAnchor)
            updateInstructionsPositions()
        }
    }
`
```

Рисунок 3.5 відстеження зміни позиції сутностей у просторі та на екрані

В методі *func session(_:didUpdate anchors:)* слідкуємо за позицією якоря зображення в просторі та оновлюємо позиції доданих сутностей відносно нього. А в *func session(_:didAdd frame:)* – уже слідкуємо на оновлюємо 2D позицію на екрані. Ми її шукаємо шляхом проектування 3D на 2D методом римітифunc project(_ point: SIMD3<Float>) -> CGPoint?. Ці точки потім використовуємо для відображення View, так що вони виходять поверх сутностей.

Коли ми закриваємо ARView, ми її очищуємо від всіх сутностей на ставимо ARSession на паузу.

3.5 Принцип роботи готового застосунку

Спочатку користувач бачить список сцен так конпку «плюс» в навігаційному меню. Якщо він натисне на неї перед ним з’явиться модульне вікно, а якому він зможе ввести початкові дані та створити блок інструкцій, які прив’язуватимуться до зображення. *(Рисунок 1 та 2)*

Після того, як користувач натисне кнопку «Save», ця сцена з’явиться в списку на початковому екрані. Після цього він може перейти до цієї сцени.

Там будуть 3 кнопки: стрілочка «назад « - повернутися до списку сцен, «сміттєвий бак» – очистити сцену він свій створених інструкцій, та «плюс» щоб

додати нову інструкцію. Знизу показуватиметься зображення, яке очікується для розпізнання. До того ж, поки не буде знайдений ARImageAnchor, функція додавання нових сутностей буде відключена. *(Рисунок 3.8, Рисунок 3.9)*

Якщо користувач обрав додавання нової інструкції, відкриється модульне вікно з двома текстовими полями та набором іконок, які можна вибрати для позначення інструкції. *(Рисунок 3.8, Рисунок 3.9)*

Назва інструкції – обов'язкове поле вводу. Якщо користувач захоче зберегти інструкцію без назви, то отримає попередження, що така дія неможлива. *(Рисунок 3.10, Рисунок 3.11)*

Створені інструкції можна редагувати та видаляти зі сцени. *(Рисунок 3,12)*

Після заповнення всіх необхідних полів, потрібно залишити мітку в просторі. Для цього з'являється зображення кружечку, яке вказує на місце, де буде поставлена інструкція. Віртуальна сутність розташовуватиметься на 25 сантиметрах від камери користувача.

Після натиску на кнопку «Place», з'явиться інструкція, яка буде змінювати своє положення відносно позиції якірного зображення. Якщо зображення було розпізнане, але більше не є в зоні видимості, все одно відбувається трекінг положення девайсу відносно місця, де було розпізнано зображення, і користувач може залишити інструкцію на деякій відстані від тої точки.

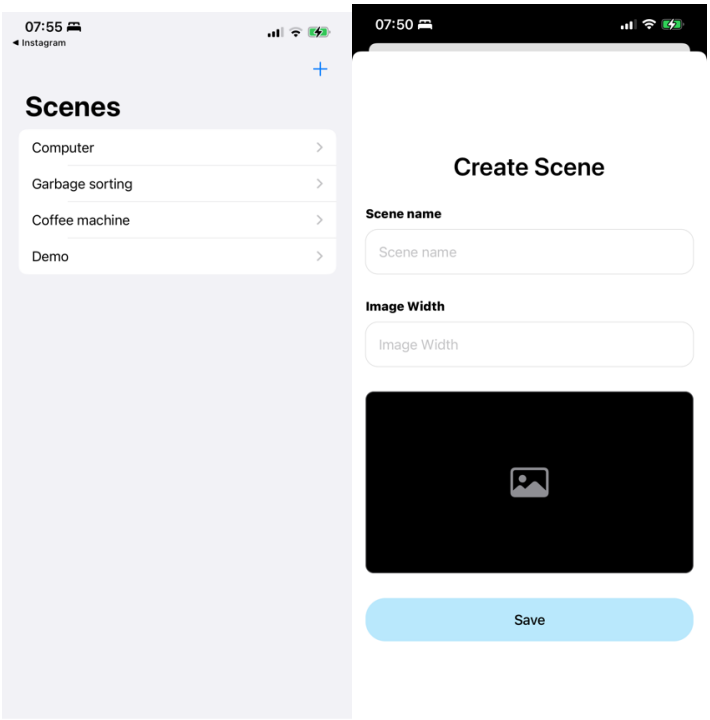


Рисунок 3.6 Список сцен

Рисунок 3.7 Форма створення нової сцени

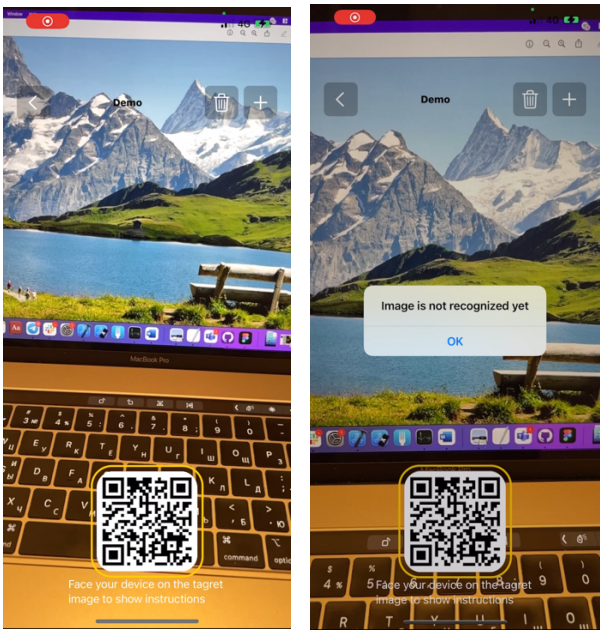


Рисунок 3.8 Форма створення нової інструкції та кнопки меню,

Рисунок 3.9 Повідомлення про некоректно введені данні

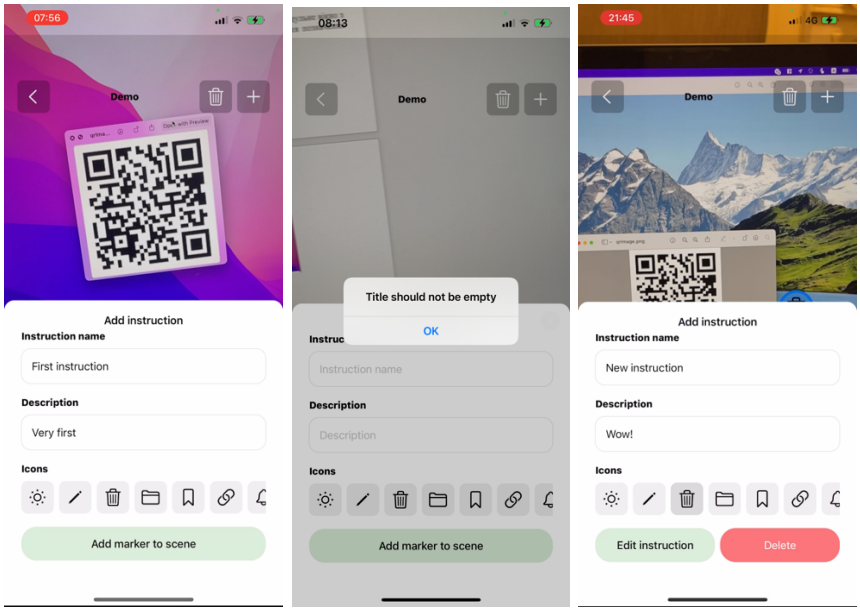


Рисунок 3.10 Форма створення нової інструкції та кнопки меню,

Рисунок 3.11 Повідомлення про некоректно введені данні

Рисунок 3.12 Форма редагування існуючої інструкції, можливіть її видалити

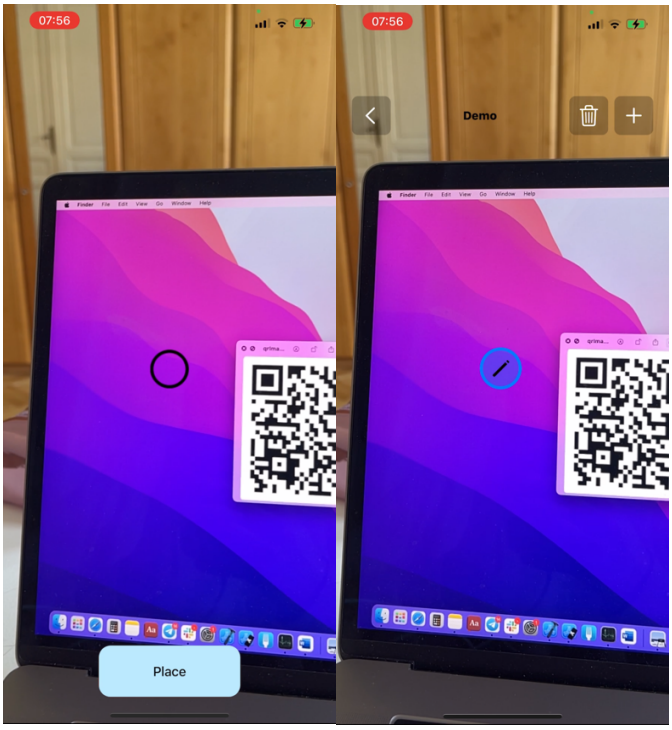


Рисунок 3.13 Екран додавання інструкції до сцени

Рисунок 3.14 Додана до сцени інструкція

3.5. Висновки до розділу 3

В розділі 3 було описане технічне завдання, використані засоби розробки застосунку та обґрунтована вибрана архітектура застосунку.

Були описані етапи розробки застосунку, такі як: створення сцени, завантаження всіх даних на сервер, створення досвіду віртуальної реальності.

В кінці розділу наведені скріншоти з роботою готового продукту.

Висновки по роботі та аналіз можливостей для подальшого розвитку застосунку

Результатом виконаної роботи стала реалізація застосунку під операційту систему iOS з використанням технологій доповненої реальності. При створенні цього програмного продукту, були виконані всі функціональні вимоги, які ставились за мету.

Досліджені технології доповненої реальності ARKit. Також був досліджений фреймворк для графічного відображення віртуального вмісту – RealityKit, новий фреймворк SwiftUI для створення графічного інтерфейсу. Провівши дослідження, були сформовані та проаналізовані недоліки та переваги кожного вибраного інструментарію.

Основною перевагою роботи з доповненої реальності на iOS є те, що обидві технології ARKit та RealityKit є нативними фреймворками Apple, через що чудово співпрацюють між собою, а ще вони надають високоякісний та потужний функціонал за простими API. Зявдяки Reality Composer – інструменту що йде разом з RealityKit фреймворком, можна «на ходу» створювати якісні моделі присвоюючи їм фізичні характеристики, аудіо, анімацію.

Не є недоліком, але ARKit підтримується лише на пристроях з операційною системою iOS, що не зважаючи на і так величезну кількість девайсів, що підтримуються фреймворком починаючи з iPhone 6s, суттєво зменшує чисельність унікальних користувачів.

Фреймворк SwiftUI має декларативний синтаксис, тому дуже легкий та зрозумілий у написанні. Однак фреймворк що достатньо молодий, тож не підтримує, або не так ефективно підтримує усі можливості його попередника – UIKit. А ще є висока ймовірність зіштовхнутися з якоюсь проблемою першим, коли ще немає відповідей на це питання в мережі.

У додатку є багато можливостей подальшого розвитку, зокрема персоналізація інструкцій, можливість їх поширення. Є багато можливостей для удосконалення користувацького інтерфейсу та досвіду користувача.

Список джерел

1. [Augmented Reality Statistics You Should Know in 2022 [Електронний ресурс] https://www.threekit.com/20-augmented-reality-statistics-you-should-know-in-2020
2. May 2022 Mobile User Statistics: Discover the Number of Phones in The World & Smartphone Penetration by Country or Region. [Електронний ресурс] https://www.bankmycell.com/blog/how-many-phones-are-in-the-world
3. Augmented Reality Principles and Practice – D.Schmaltieg, T. Höllerer – c 3
4. Augmented Reality Principles and Practice – D.Schmaltieg, T. Höllerer – c 4
5. Building in AR: Some basics and where to start [Електронний ресурс] https://arvrjourney.com/building-ar-where-to-start-6611b3dca51f
6. DoF, FoV, Anchors, and Markers: Demystifying Common AR Terms [Електронний ресурс] https://learningsolutionsmag.com/articles/dof-fov-anchors-and-markers-demystifying-common-ar-terms
7. What are the different types of Augmented Reality? [Електронний ресурс] https://softtek.eu/en/tech-magazine-en/user-experience-en/what-are-the-different-types-of-augmented-reality/
8. What Is SLAM? [Електронний ресурс] https://www.mathworks.com/discovery/slam.html
9. ARKit [Електронний ресурс] https://developer.apple.com/augmented-reality/arkit/
10. Introducing ARKit: Augmented Reality for iOS [Електронний ресурс] https://developer.apple.com/videos/play/wwdc2017/602/
11. ARSesssion [Електронний ресурс] https://developer.apple.com/documentation/arkit/arsession
12. ARAnchor [Електронний ресурс] https://developer.apple.com/documentation/arkit/aranchor
13. Ray casting [Електронний ресурс] https://ru.wikipedia.org/wiki/Ray_casting

14. Introduction to RealityKit on iOS— Entities, Gestures, and Ray Casting [Електронний ресурс] https://www.iosdevie.com/p/introduction-to-realitykit-on-ios?s=r
15. Martering ARKit – Jeyven Nhan, p 37
16. What is SwiftUI? [Електронний ресурс] https://codewithchris.com/swiftui/what-is-swiftui/
17. What is Firebase? [Електронний ресурс] https://www.educative.io/edpresso/what-is-firebase
18. Package Manager [Електронний ресурс] https://www.swift.org/package-manager/
19. MVVM in iOS Swift: Using the SwiftUI view model [Електронний ресурс] https://sevenpeakssoftware.com/mvvm-in-ios-with-swift-ui-article/