НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет електроніки
(повна назва інституту/факультету)

Кафедра звукотехніки та реєстрації інформації
(повна назва кафедри)

«На правах рукопису»
УДК 004.422.81

«До захисту допущено»

Завідувач кафедри
_____ Г.Г.Власюк
(підпис) (ініціали, прізвище)
“___” _____2019 р.

Магістерська дисертація

спеціальність 171 Електроніка
(код і назва спеціальності)

на тему: «Використання технологій доповненої реальності для потреб мультимедіа»

Виконав: студент VI курсу, групи ДВ-81мп
(шифр групи)

Курило Данило Назарович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник асистент, к.т.н., Романюк М. І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

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

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

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

Інститут (факультет) Факультет електроніки
(повна назва)

Кафедра звукотехніки та реєстрації інформації
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою

Спеціальність (освітня програма) 171 Електроніка

(Електронні системи мультимедіа та засоби Інтернету речей)

(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Г.Г.Власюк
(підпис) (ініціали, прізвище)
« » грудня 2019 р.

ЗАВДАННЯ

на магістерську дисертацію студенту

Курило Данилу Назаровичу

(прізвище, ім'я, по батькові)

1. Тема дисертації «Використання технологій доповненої реальності для потреб мультимедіа»

науковий керівник дисертації Романюк Маргарита Ігорівна, к.т.н.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від « 07 » листопада 2019 р. № 3859-с

2. Строк подання студентом дисертації 09.12.2019 р.

3. Об'єкт дослідження: технології доповненої реальності мобільних застосунків

4. Предмет дослідження (Початкові дані – для магістерської дисертації за освітньо-професійною програмою): методи та технології розроблення мобільного застосунку для створення доповненої реальності, середовище розробки програмного забезпечення Xcode, бібліотека TorchKit, бібліотека SwiftEntryKit, мобільний телефон на базі iOS, комп'ютер на базі macOS.

5. Перелік завдань, які потрібно розробити: 1) Проаналізувати існуючі рішення інтеграції доповненої реальності у мобільні застосунки; 2) Визначити програмні та апаратні засоби створення доповненої реальності; 3) Проаналізувати Torch SDK та його можливості та засоби інтеграції у застосунки на базі операційної системи iOS. 4) Створити застосунок для Національного художнього музею України використовуючи фреймворк Torch SDK.

6. Перелік графічного (ілюстративного) матеріалу: 1) 36 рис, 23 табл., 1 презентація, 8 слайдів.

7. Орієнтовний перелік публікацій: 1.) Огляд методів взаємодії зі смарт-окулярами // XV Міжнародна науково-практична інтернет-конференція «РОЗВИТОК СУЧАСНОЇ НАУКИ:ТЕОРІЯ, ПРАКТИКА, ІННОВАЦІЇ», 2019 р.,-С.51-53 2.) Використання штучного інтелекту в музичній індустрії // ІІ ВСЕУКРАЇНСЬКА НАУКОВО-ТЕХНІЧНА КОНФЕРЕНЦІЯ «СУЧАСНІ ТЕХНОЛОГІЇ КІНО ТА АУДІОВІЗУАЛЬНИХ СИСТЕМ», 2019 р., -С.19-21.

8. Дата видачі завдання 10. 09. 2018 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Написання першого розділу: Аналітичний огляд існуючих AR рішень	15.03.2019	
2	Написання другого розділу: Особливості застосування середовища Torch для створення застосунку з AR	30.05.2019	
3	Написання третього розділу: Практична реалізація проекту Torch для імпортування в застосунок	10.10.2019	
4	Підготовка матеріалів до друку та оформлення пояснювальної записки	30.11.2019	
5	Підготовка та оформлення презентації для доповіді	03.12.2019	

Студент _____ Д.Н. Курило
(підпис) (ініціали, прізвище)

Науковий керівник _____ М.І. Романюк
(підпис) (ініціали, прізвище)

УДК 004.422.81

РЕФЕРАТ

Магістерська дисертація: 97 с., 36 рис., 23 табл., 2 дод., 15 джерел.

ДОПОВНЕНА РЕАЛЬНІСТЬ, МОБІЛЬНИЙ ЗАСТОСУНОК, МОВА ПРОГРАМУВАННЯ SWIFT, ПРОГРАМНА ПЛАТФОРМА TORCH SDK

Актуальність дослідження. Число мобільних застосунків, що використовують технології доповненої реальності стрімко росте. Про актуальність обраної теми свідчать переваги, які надає використання технологій доповненої реальності в навігації та проведені музейних екскурсій, а саме:

- підвищення ступеня цікавості екскурсій з використанням доповненої реальності в порівнянні з традиційними екскурсіями;

- підвищення ступеня автономності та самостійності користувачів при проходженні екскурсій завдяки відсутності необхідності формувати групи та використання послуг куратора;

- підвищення охоплення слабочуючих або погано бачущих груп людей завдяки подання інформації в текстовому виді та наявності аудіосупровіду.

Метою дослідження є створення мобільного застосунку з доповненою реальністю на базі операційної системи iOS для відвідувачів Національного художнього музею України.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати існуючі технології доповненої реальності у мобільних застосунках;

- обрати технологію, яка найкраще реалізує потреби музею;

- розробити застосунок на базі операційної системи iOS на мові програмування Swift.

Об’єкт дослідження – технології доповненої реальності мобільних застосунків та способи створення застосунків на базі операційної системи iOS.

Предмет дослідження – методи та технології розроблення мобільного застосунку для створення доповненої реальності.

Методи дослідження – Теоретично-аналітичний і практичний аналіз технологій реалізації доповненої реальності: мова програмування Swift, бібліотека Torch SDK, середовище Xcode.

Наукова новизна отриманих результатів: запропонований застосунок який вперше використовує бібліотеку Torch SDK.

Практична значення одержаних результатів: запропоновано та створено застосунок, що надає можливість музейним співробітникам редагувати та додавати інформацію про відстежувані об’єкти, без необхідності оновлювати застосунок, та писати програмний код.

Апробація результатів дисертації: Презентація застосунку у UNIT.City.

SUMMARY

Master’s dissertation: 95 p., 36 pic., 23 tabl., 15 sources, 2 supplements.

AUGMENTED REALITY, MOBILE APPLICATION, SWIFT PROGRAMMING LANGUAGE, TORCH SDK

The relevance of the topic of the work is that augmented reality as a method of delivering information is widespread in various fields, in particular in the development of mobile applications.

The subject of the study is augmented reality mobile application technologies and ways to build applications based on the iOS operating system.

The purpose of the study is to create a mobile app with augmented reality based on the iOS operating system for visitors of the National Art Museum of Ukraine.

To achieve this goal, the following tasks must be performed:

- analyze existing augmented reality technologies in mobile applications.
- choose the technology that best meets the needs of the museum.
- create an application based on the iOS operating system in Swift programming language.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

AR – Augmented reality (доповнена реальність);

VR – Virtual reality (штучна, віртуальна реальність);

API – Application Programming Interface (інтерфейс прикладного програмування);

JSON – JavaScript Object Notation (запис об’єктів JavaScript)

USD – Universal Scene Description (універсальний опис сцени)

USDZ – Universal Scene Description Zip

UI – User Interface (користувацький інтерфейс)

UWP – Universal Windows Platform (Універсальна платформа Windows)

SDK – Software Development Kit (набір із засобів розробки, утиліт і документації, який дозволяє програмістам створювати прикладні програми за визначеною технологією)

ВСТУП

Термін доповненої реальності (augmented reality, AR) імовірно був запропонований працюючим на корпорацію Boeing дослідником Томом Коделом в 1990 р. У 1992 р. Луї Розенбург із дослідницької лабораторії Армстронгу США створив першу реальну систему доповненої реальності під назвою "Віртуальний світильник". Це робототехнічна система, яка розміщує інформацію поверх робочого середовища працівників, щоб допомогти досягти більшої ефективності [1].

Доповнена реальність є візуальним доповненням реального світу, який отримується шляхом проектування і виведення будь-яких віртуальних, уявних об’єктів на даний простір (на екран комп’ютера, планшета, телефону). На сьогодні основною технологією доповненої реальності є отримання зображення з камери, його оброблення алгоритмами розпізнавання образів, з подальшим накладенням зображення. Технології доповненої реальності з кожним днем все міцніше інтегруються у сучасному світі. Ця технологія швидко стала більш життєздатною для комерційних та дослідницьких проектів за останнє десятиліття завдяки перевазі пристроїв, що встановлюються на голові (HMD - Head Mounted Display) та розумних пристроїв, таких як телефони, планшети та портативні ігрові приставки, які тепер невід'ємно вбудовані у повсякденне життя. Це зменшило основну проблему розгортання додатка з додатковою реальністю, оскільки для використання технології більше не потрібно спеціалізоване обладнання, натомість користувачі можуть керувати системою на власних пристроях. При зменшенні апаратного бар'єру доповнена реальність почала використовуватись у багатьох сферах: від ігор до освіти, медицини і промисловості.

Введення в 2017 році бібліотек ARKit від Apple і ARCore від Google стандартизувало інструменти розробки та створення програм для мобільних пристроїв з доповненою реальністю, що принесло майже удвічі більше мобільних пристроїв з підтримкою AR та втричі збільшило кількість активних користувачів протягом 1,5 років. Залучивши AR до масової аудиторії користувачів мобільних

пристроїв, Apple забезпечила своє лідерство на ринку AR, коли презентувала ARKit 2.0 на WWDC 2018, а потім ARKit 3.0 на WWDC 2019. Що стосується технології, то впроваджені рішення, дозволили розмістити мобільну AR в одній категорії з AR на базі існуючої гарнітури.

Доповнена реальність робить можливим впровадження синтезованих об’єктів у природні сцени, чим відрізняється від "віртуальної реальності", яка є повністю штучно синтезованим світом (відеорядом).

Доповнена реальність ассоціюється зі складними технологіями та ресурсоємними процесами розробки. Саме тому деякі компанії не розглядають доповнену реальність як спосіб залучення нових клієнтів та розвинення своєї цифрової актуальності. Тому у роботі розглядаються існуючі засоби створення доповненої реальності, та пропонується рішення для створення екскурсійних застосунків з доповненою реальністю за допомогою бібліотеки Torch SDK. У запропонованого рішення є потенціал розвиватись до універсальної платформи для створення екскурсійних AR застосунків без навичок програмування.

1 АНАЛІТИЧНИЙ ОГЛЯД ІСНУЮЧИХ AR РІШЕНЬ

1.1. Основи та принципи роботи технологій доповненої реальності

Важливо розуміти відмінності між доповненою реальністю і змішаною реальністю. У широкому сенсі доповнена реальність являє собою процес перегляду реального світу і віртуальних об’єктів одночасно, де віртуальна інформація накладається, вирівнюється та інтегрується в фізичному світі. Доповнена реальність знаходиться в безперервному діапазоні інтерфейсів від «реальності» до віртуальної реальності «повного занурення».

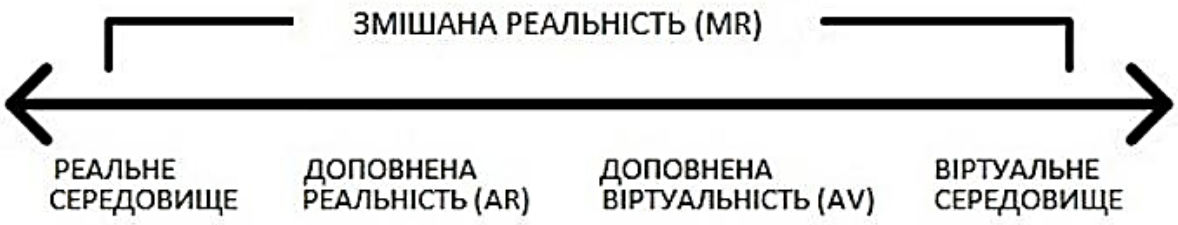


Рисунок 1.1 – Діапазон технологій змішаної реальності

Істотною відмінністю доповненої реальності від віртуальної є збереження фізичного світу як контексту, в якому представлені віртуальні об’єкти і з яким вони взаємодіють. Віртуальна реальність повністю абстрагується від фізичного світу, щоб помістити користувача повністю у віртуальний світ. Віртуальна реальність використовує спеціальні позиційні трекери з дисплеями (окуляри віртуальної реальності), які динамічно оновлюють видимий користувачем простір у віртуальному середовищі. Доповнена реальність повністю змінює цю парадигму, і в підсумку віртуальні об’єкти розміщуються в реальному оточенні користувача.

Таким чином, доповнена реальність – це технології, що дозволяють доповнювати зображення реальних об’єктів різними об’єктами комп’ютерної графіки, а також поєднувати зображення, отримані від різних джерел комп’ютерного середовища: відеокамер, акселерометрів, компасів і т.д. Схема середовища доповненої реальності представлена на рис. 2.2. На відміну від «віртуальної реальності», яка передбачає повністю штучний синтезований світ

(відеоряд), доповнена реальність припускає інтеграцію віртуальних об’єктів у природні відеосцени.

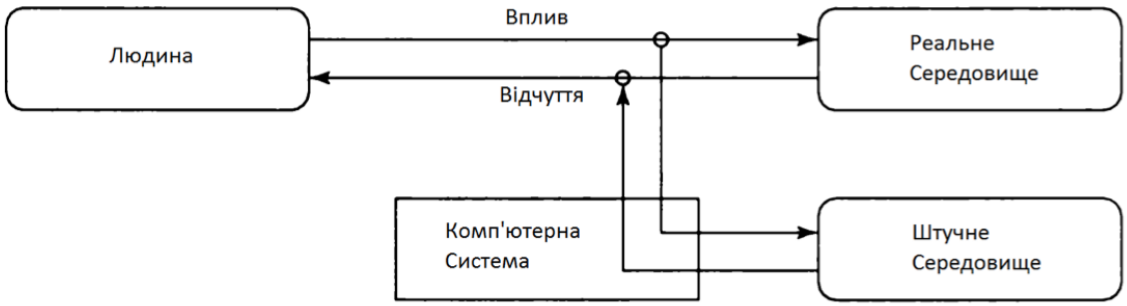


Рисунок 1.2 – Схема середовища доповненої реальності

Людина отримує уявлення про навколишній простір за допомогою великого набору органів сприйняття навколишньої інформації. Система доповненої реальності, що є посередником між людиною і реальністю, повинна створювати сигнал для одного з таких органів. Таким чином, за типом подання інформації системи доповненої реальності бувають:

Візуальні. В їх основі лежить зорове сприйняття людини. Завдання таких систем – створити зображення, яке буде використано людиною. Оскільки зображення для людини є більш інформативним і зрозумілим, такий вид систем є більш поширеним.

Аудіо. Такі системи орієнтовані на слухове сприйняття. Найчастіше такі системи використовуються в навігації. Наприклад, вони видають спеціальні сигнали, коли людина досягає певного місця. Можливе використання стереоскопічного ефекту, що дозволяє людині йти в потрібному напрямку, орієнтуючись на джерело звуку.

Аудіовізуальні. Це комбінація двох попередніх типів, однак, аудіоінформація в них має лише допоміжний характер.

Системи доповненої реальності завжди потребують інформації, одержуваної з навколишнього середовища. Саме на основі цих даних будуються віртуальні об’єкти. Кожна з таких систем володіє певним набором сенсорів – пристроїв, що

дозволяють збирати інформацію з навколишнього середовища: звукові і електромагнітні коливання, прискорення і т.д. Для класифікації має сенс розділяти сенсори не по типам реєстрованих фізичних величин, а за їх призначенням, оскільки подібні за своєю природою сигнали можуть нести різну інформацію. За типом сенсорів можна виділити наступні системи [2]:

Геопозиційні. Орієнтуються, перш за все, на сигнали систем позиціонування GPS. На застосунок до приймачів таких сигналів геопозиційні системи можуть використовувати компас і акселерометр для визначення кута повороту відносно вертикалі і азимута.

Оптичні. Такі системи обробляють зображення, отримане з камери, яка переміщається разом з системою або незалежно від неї.

Системи доповненої реальності можна розрізняти за ступенем взаємодії з користувачем. У деяких системах користувач грає пасивну роль, він лише спостерігає за реакцією системи на зміни в навколишньому середовищі. Інші ж системи вимагають активного втручання користувача – він може управляти як роботою самої системи, для досягнення результатів, так і змінювати віртуальні об’єкти. За цією ознакою системи діляться на:

Автономні. Вони не вимагають втручання користувача. Завдання таких систем зводиться до надання інформації про об’єкти. Наприклад, подібні системи можуть аналізувати об’єкти, що знаходяться в полі зору людини і видавати довідкову інформацію про них. Також системи такого типу використовуються в медицині.

Інтерактивні. Такі системи засновані на взаємодії з користувачем. На різні дії користувач отримує різну відповідь. У подібних системах необхідно мати пристрій введення інформації. В якості такого пристрою може застосовуватися сенсорний екран мобільного телефону, планшет або спеціальний маніпулятор. Вибір пристроїв введення залежить від специфіки системи. У разі простих дій з віртуальним об’єктом, достатньо простого вказівного пристрою. Якщо ж необхідна імітація будь-яких реальних процесів і виконання складних маніпуляцій з

об’єктами використовуються спеціальні маніпулятори, які мають різну кількість ступенів свободи.

Інтерактивність виражається в різному ступені. Бувають системи, що дозволяють користувачеві активно змінювати віртуальне середовище. Зазвичай це системи-симулятори будь-яких реальних дій. Вони використовуються у разі, коли використання реальних об’єктів неможливо, наприклад, спеціалізовані медичні тренажери, що дозволяють початківцям лікарям відпрацьовувати необхідні навички.

Існують інші системи, де користувачеві не потрібно змінювати віртуальне середовище. Замість цього користувач обирає, які віртуальні об’єкти він хоче побачити. Користувач також має можливість маніпулювати віртуальними об’єктами, але не на рівні структури, а на рівні відображення, тобто застосовувати, наприклад афінні перетворення типу повороту, переміщення і т.д. До даної групи можна віднести різні архітектурні системи, що дозволяють побачити, як впишеться в реально існуючу обстановку нова споруда або його частина, а також навігаційні та геоінформаційні системи. Подібні системи можуть показувати частини об’єктів інтересу, приховані іншими будівлями, додаткову інформацію про обрані об’єкти, тощо.

За ступенем мобільності системи доповненої реальності можна класифікувати як:

Стаціонарні. Системи цього типу призначені для роботи в фіксованому місці; переміщення таких систем означає часткове або повне припинення їх працездатності.

Мобільні. Системи цього типу можуть без зусиль переміщатися; часто таке переміщення і лежить в основі виконуваної ними функції.

Належність до того чи іншого типу визначається функціями системи. Так, симулятор хірургічного столу не повинен бути мобільним, оскільки його завдання – відтворити для людини спеціальні умови, максимально наближені до реальних. У той же час навігаційна система повинна бути якомога більш мобільною, щоб вона

могла переміщатися разом з транспортним засобом або людиною, не створюючи застосункових витрат на її переміщення.

1.2. Програмні засоби створення AR

1.2.1 ARKit

5 червня 2017 року Apple представив ARKit [3], нову програмну платформу (фреймворк), що дозволяє користувачам легше створювати застосунки доповненої реальності для iPhone та iPad. ARKit – це API високого рівня, який забезпечує простий інтерфейс з потужним набором функцій. ARKit працює на процесорах Apple A9 та A10, тобто на нових пристроях iOS, починаючи з iPhone 6S. Особливості ARKit:

1) *Відстеження*: трекінг – це основна функція ARKit. ARKit використовує Visual Inertial Odometry (VIO) для точного відстеження навколишнього світу. Візуальна одометрія означає оцінку тривимірної пози (переміщення + орієнтація в просторі) рухомої камери відносно її вхідного положення з використанням візуальних особливостей. VIO з'єднує дані датчиків камери з даними CoreMotion. Ці два входи дозволяють пристрою відчувати, як він рухається в приміщенні з високим ступенем точності без додаткового калібрування. Що ще важливіше, не потрібні зовнішні налаштування, не потрібні попередні знання для навколишнього середовища, а також не потрібні застосункові датчики.

2) *Розуміння сцени* – це здатність визначати атрибути чи властивості довкілля навколо пристрою. За допомогою пристрою ARKit може аналізувати сцену, представлену переглядом камери, та знаходити горизонтальні площини в кімнаті. ARKit також використовує датчик камери для оцінки загальної кількості світла, наявного в сцені, і застосовує правильну кількість освітлення до віртуальних об'єктів. Функціональність хіт-тестування забезпечує перетин топології реального світу, щоб віртуальні об'єкти можна було розмістити у фізичному світі.

3) *Візуалізація*: ARKit забезпечує постійний потік зображень з камери, залучаючи інформацію та розуміння сцен, які можуть бути введені в будь-який

візуалізатор, включаючи SceneKit, Metal, SpriteKit та сторонні інструменти, такі як Unity та Unreal Engine.

1.2.2 ARCore

ARCore – платформа Google для створення доповненої реальності. Використовуючи різні API, ARCore дозволяє телефону визначати своє оточення, «розуміти» світ та взаємодіяти з інформацією. Деякі API доступні для Android та iOS, щоб отримати спільний доступ до AR. У ARCore використовують три ключові можливості для інтеграції віртуального контенту з реальним світом, що спостерігається через камеру телефону:

- Відстеження руху дозволяє визначати та відстежувати координати телефона у просторі.
- Координати простору дозволяють телефону визначати розміри та розташування всіх типів поверхонь: горизонтальних, вертикальних і кутових поверхонь, таких як земля, журнальний столик або стіни.
- Оцінка світла дозволяє алгоритмам дати оцінку поточним умовам освітлення навколишнього середовища.

Розуміння наступних основоположних концепцій ARCore має вирішальне значення для того, щоб почати створювати застосунки, завдяки яким віртуальний вміст може опинитися на реальних поверхнях або бути приєднаним до реальних локацій [4]:

Anchor (опорна точка): описує фіксовану локацію та орієнтацію в реальному світі. Щоб залишитися у фіксованому місці у фізичному просторі, числовий опис цієї позиції буде оновлятись з поліпшенням розуміння простору ARCore.

HitResult: визначає перетин між променем та оціненою геометрією реального світу.

Plane (площина): описує поточні найкращі знання планарної поверхні реального світу.

PlaneHitResult: визначає перетин між геометрією променя і площини.

PointCloud: містить набір спостережуваних 3D точок та значення достовірності.

Pose (поза): являє собою незмінне жорстке переміщення від одного координатного кадру до іншого. Він завжди описує переміщення від локального координатного кадру об’єкта до світового координатного кадру. Тобто, пози з API ARCore можна вважати еквівалентними матрицям моделі OpenGL. Трансформація визначається за допомогою обертання кватерніона щодо початку координат з подальшим переміщенням.

Session: керує станом системи AR та обробляє життєвий цикл сеансу. Це головна точка входу в ARCore API, що дозволяє користувачеві створювати сеанс, налаштовувати його, запускати/зупиняти його і, головне, отримувати кадри, які дозволяють отримати доступ до зображення камери та позу пристроя.

Світова система координат: Оскільки розуміння ARCore щодо навколишнього середовища змінюється, API підлаштовує свою модель світу, щоб підтримувати стабільність. Коли це стається, числове розташування (координати) камери та anchor’а можуть суттєво змінитися, щоб підтримувати відповідні відносні положення фізичних розташувань, які вони представляють. Ці зміни означають, що кожен кадр слід вважати таким, що знаходиться у абсолютно унікальній світовій координатній системі. Чисельні координати anchor-ів і камери ніколи не повинні використовуватися поза кадром візуалізації, під час якого вони були отримані. Якщо позицію потрібно розглядати за рамками одного кадру візуалізації, слід створити або anchor або використовувати положення відносно існуючого anchor-у поблизу.

1.2.3. Vuforia

Vuforia Engine [5] - це найпоширеніша платформа для розробки AR застосунків, яка підтримує провідні телефони, планшети та окуляри. Розробники можуть легко додати розширені функціональні можливості комп’ютерного зору

для застосунків Android, iOS та UWP (Universal Windows Platform), щоб створити AR, який реалістично взаємодіє з об’єктами та оточенням.

Можливості розпізнавання та відстеження Vuforia Engine можна використовувати на різних зображеннях і об’єктах:

- *Цілі у вигляді 3D моделей*: Дозволяє розпізнавати об’єкти за формою за допомогою вже існуючих 3D-моделей. Дозволяє розміщувати AR на широкому спектрі предметів, таких як промислове обладнання, транспортні засоби, іграшки та побутова техніка.
- *Зображення*: Дозволяє додавати AR до плоских зображень, таких як друковані носії та упаковки продукту.
- *Об’єкти*: створюються шляхом сканування об’єкта. Є хорошим варіантом для іграшок та інших виробів з детальними поверхнями та стійкою формою.
- *Багатоцільові об’єкти*: створюються за допомогою декількох цільових зображень і можуть бути розміщені у вигляді базових геометричних фігур або в будь-якому довільному розташуванні плоских поверхонь.
- *Циліндри*: Дозволяє розпізнавати зображення, загорнуті на предмети приблизно циліндричної форми (наприклад, пляшки для напоїв, чашки для кави, банки з напоями).
- *VuMarks*: це спеціалізовані маркери, які можуть кодувати різні формати даних. Вони підтримують унікальну ідентифікацію та трекінг програм AR.
- *Зовнішня камера*: Дозволяє отримувати доступ до відеоданих із камери поза тією, що знаходиться в телефоні чи планшеті, всередині AR сесії.
- *Земна площина*: Дозволяє розміщувати вміст на горизонтальних поверхнях навколишнього середовища, таких як столи та підлоги.

Vuforia Fusion призначений для забезпечення найкращого можливого досвіду AR на широкому спектрі пристроїв. Цей фреймворк використовує можливості пристрою та з’єднує їх за допомогою функцій Vuforia Engine, що дозволяє розробникам працювати тільки з API Vuforia для оптимального використання AR.

Vuforia Engine також може розпізнавати та відстежувати широкий спектр 3D-об’єктів. Розпізнавання об’єктів дозволяє створювати цілі для трекінгу шляхом сканування фізичних об’єктів. Це дозволяє створювати застосунки, які розпізнають і відстежують складні об’єкти.

Служба розпізнавання у хмарі допомагає розпізнавати великий набір зображень, або використовується якщо база даних часто оновлюється. API веб-служб Vuforia дозволяє ефективно керувати цими великими базами зображень у хмарі та дозволяє автоматизувати робочі процеси шляхом прямої інтеграції у системи управління вмістом.

Драйвери Vuforia Engine дозволяють розробникам надавати та споживати дані із зовнішніх систем через Vuforia Engine. Цей фреймворк забезпечує доступ до функцій зовнішньої камери. Зовнішня камера визначає всі особливості, необхідні Vuforia Engine для доступу до зовнішніх джерел зображення.

1.2.4 Wikitude

Wikitude [6] включає розпізнавання зображень та трекінг, підтримує 3D-рендерінг моделі з накладанням відео та забезпечує геопозиційний AR. Wikitude SDK поєднує в собі можливості геопозиційного розпізнавання та розпізнавання зображень, щоб забезпечити гібридне відстеження. Цей фреймворк будується на веб-технологіях (HTML, JavaScript та CSS), які дозволяють писати кросплатформені застосунки доповненої реальності, сформовані як об’єкти ARchitect і в основному являють собою звичайні сторінки HTML, які можуть використовувати API ARchitect для створення об’єктів у доповненій реальності. Фреймворк Wikitude SDK можна інтегрувати в застосунки, додавши в інтерфейс користувача компонент подання, який називається ARchitectView. Wikitude SDK є комерційним рішенням, але він також доступний у вигляді пробної версії з деякими обмеженнями, такими як логотип Wikitude у режимі перегляду камери тощо. SDK Wikitude зараз доступний для платформ Android та iOS. Браузер Wikitude AR спочатку був випущений на платформі Android, а потім поширився на Blackberry,

Bada, Windows Phone та iOS. Wikitude також надає студію Wikitude, що полегшує процедуру розробки, де не потрібні навички програмування, а застосунок можна створити простим перетягуванням об’єктів на екран студії.

1.2.5 Порівняння розглянутих фреймворків

Усі перелічені SDK мають свої переваги та недоліки, нижче представлено у таблицях 1.1, 1.2,1.3 з порівняння основних характеристик.

Таблиця 1.1 – Порівняння типів ліцензій фреймворків

AR SDK		ARKit	ARCore	Vuforia	Wikitude
Тип					
Ліцензія	Open-source	Ні	Ні	Ні	Ні
	Безкоштовна	Так	Так	Так	Так
	Комерційна	Ні	Ні	Так	Так

Таблиця 1.2 – Порівняння платформ, що підтримують фреймворки

AR SDK		ARKit	ARCore	Vuforia	Wikitude
Тип					
Платформа	iOS	Так	Так	Так	Так
	Android	Ні	Так	Так	Так
	Windows	Ні	Ні	Так	Так

Код жодної з платформ не зберігається у відкритому доступі. Усі платформи безкоштовні для використання, окрім Vuforia та Wikitude – для їх використання у комерційних цілях потрібна ліцензія.

Vuforia та Wikitude підтримують усі популярні операційні системи, ARCore – тільки Android та нещодавно iOS. ARKit залишається доступним ексклюзивно на iOS.

Таблиця 1.3 – Порівняння способів трекінгу різних платформ

AR SDK		ARKit	ARCore	Vuforia	Wikitude
Тип					
Трекінг	GPS	Так	Так	Ні	Так
	ІВП	Так	Так	Ні	Так
	Обличчя	Так	Так	Ні	Ні
	Ключові точки	Так	Так	Так	Так
	3D моделі	Так	Так	Так	Ні
	Інше	Motion capture, виявлення та трекінг людей	Опорні точки, що зберігаються на хмарному сервері	Розширений трекінг, локалізоване виявлення оклюзії	Гібридний трекінг, розширений трекінг

ARKit та ARCore підтримують майже усі відомі способи трекінгу, ARKit розширює свій функціонал додаванням захвату рухів та відстеженню людей. Vuforia та Wikitude дещо позбавлені ключових можливостей, але якщо в застосунку вони не потрібні – ці фреймворки все ще мають свої переваги.

1.3 Апаратні засоби створення AR

1.3.1 Підтримка пристроїв iOS

З виходом iOS 11 та ARkit у 2017, Apple анонсувала підтримку AR на пристроях на базі процесору A9 та пізніше, а саме:

- iPhone 6s та 6s Plus;
- iPhone 7 та 7 Plus;
- iPhone SE;
- iPad Pro (9.7, 10.5 або 12.9) – першого та другого поколінь;
- iPad (2017);

- iPhone 8 та 8 Plus;
- iPhone X.

Пізніші моделі пристроїв також підтримують ARKit

- iPhone XS, XR та XS Max;
- iPhone 11, 11 Pro, та 11 Pro Max

З виходом ARKit 2 у 2018 технічні вимоги не змінилися.

ARkit 3, що дає доступ до таких можливостей як розпізнавання людей та розміщення 3D об’єктів відносно них (People occlusion), захоплення руху людини у реальному часі, та віддзеркалення цих рухів на 3D персонажів (motion capture) та ін. доступний тільки на пристроях з чіпом (процесором) A12. Тобто, можливості ARkit 3 підтримують лише (див. рис. 1.3):

- iPhone XS, XR та XS Max
- iPhone 11, 11 Pro, та 11 Pro Max
- обидва iPad 2018-ого року
-

Рисунок 1.3 – підтримка ARkit 3

ARkit 3 підтримується лише новими пристроями, тому що на використання таких функцій як motion capture та people occlusion потрібна висока обчислювальна

спроможність, яку надають тільки нові процесори. Переваги процесору Apple A12 Bionic:

- Має на 4 ядра більше;
- Підтримка більшої кількості ОЗУ (12 проти 4 ГБ);
- В 2,3 рази вище пропускна здатність пам’яті (34,1 проти 14,9 Гбіт/с);
- Менший розмір транзистора (7 проти 14 нанометрів);
- На 35% вище частота процесора (2490 проти 1850 МГц);
- Більш нова версія DirectX (12 проти 11);
- Краща архітектура набору команд.

1.3.2 Підтримка пристроїв Android

Для того, щоб Android пристрій мав підтримку AR, він має пройти сертифікацію Google [7]. Сертифікація важлива, оскільки Google прагне, щоб користувачі мали гарний досвід роботи з програмами AR. В першу чергу це пов’язано з чутливим відстеженням руху, що здійснюється за допомогою комбінування зображення камери та входу датчика руху, щоб визначити, як пристрій користувача рухається в реальному світі.

Щоб сертифікувати кожен пристрій, перевіряється якість камери, датчиків руху та архітектуру дизайну електронних компонентів, щоб переконатися, що все працює як слід. Також пристрій повинен мати достатньо потужний процесор, який інтегрується в апаратну конструкцію, щоб забезпечити хорошу продуктивність та ефективні розрахунки в режимі реального часу.

Перевага Android полягає у великій різноманітності пристроїв, доступних у всьому світі. Google постійно працює з виробниками, щоб переконатися, що їх обладнання та дизайн відповідають цим вимогам. У той же час Google працює внутрішньо, щоб переконатися, що ARCore добре інтегрується з кожною моделлю, яку Google сертифікує, щоб забезпечити хороший досвід для користувачів.

Базові вимоги ARCore:

- Android 7.0 або новішої версії (для деяких моделей потрібні новіші версії)
- Пристрій, який оригінально постачався з маркетом Google Play Store
- Доступ до Інтернету, щоб встановити або оновити служби Google Play для AR

1.3.3 Способи переміщення об’єктів у просторі

Ступені свободи або DOF (Degrees of freedom) – це способи переміщення об’єкта в просторі [8]. В тримірному просторі існує 6 ступенів свободи. 6 DOF можна розділити на 2 категорії, обертальні рухи та поступальні рухи. У кожній категорії є 3 DOF. Для справжнього досвіду AR потрібна підтримка як відстеження орієнтації (обертання), так і позиційного відстеження (переміщення).

3 обертальних рухи – це рух у поперечній осі (pitch), вертикальній (yaw) та повздожній (roll) осях. Ці рухи відстежуються більшістю вбудованих датчиків у пристрій. При нахилі чи обертанні пристрою, він ідентифікує рухи і відповідно змінює відображення об’єктів. Обертальні рухи відслідковуються ІВП (інерційним вимірювальним пристроєм). Інерційні вимірювальні пристрої складаються з акселерометра, гіроскопа та магнітометра. Ці ІВП вимірюють швидкість, орієнтацію та гравітаційні сили пристрою для обчислення обертальної орієнтації та руху. Ці три ІВП часто позначаються як "9 DOF". 9 DOF обчислюються шляхом додавання 3 DOF, виявлених кожним ІВП. Насправді акселерометр, гіроскоп і магнітометр вимірюють однакові 3 DOF: pitch, yaw та roll.

3 рухи переміщення – вліво/вправо, вперед/назад і вгору/вниз. Ці рухи зазвичай відслідковуються зовнішньою камерою або іншими датчиками. Уміння відстежувати поступальні рухи необхідні для позиційного відстеження, здатність визначати абсолютне положення об’єкта в 3D-середовищі.

1.3.4 Система SLAM

SLAM – Simultaneous Localization and Mapping (одночасна локалізація та картографування) – це технологія, яка представляє фізичний світ за допомогою характеристичних точок [9]. Це дає змогу застосункам AR розпізнавати 3D-об’єкти та сцени, а також миттєво відстежувати світ та накладати цифрові інтерактивні доповнення.

При першому запуску застосунку AR за допомогою Google ARCore, Apple ARKit або Microsoft Mixed Reality, система не володіє інформацією про навколишнє середовище. Вона починає обробляти дані з різних джерел – переважно з камери. Для підвищення точності пристрій поєднує дані інших корисних датчиків, таких як акселерометр та гіроскоп.

На основі цих даних алгоритм має дві цілі:

- Побудувати карту навколишнього середовища;
- Знайти пристрій у цьому середовищі.

Якщо існує свобода розміщувати маячки у відомих місцях, програмі просто потрібно розрахувати відстані, і це дає можливість точного відстеження місця знаходження об’єкту (користувача). Для інших програм GPS може бути досить корисним. Однак мобільна доповнена реальність зазвичай не використовує маячки. Також GPS не є достатньо точним, особливо в приміщеннях.

У ідеальній ситуації пристрій має досконалу інформацію про точне розташування усіх об’єктів. Сюди входить розташування маячка (1), а також розташування робота (2) (див. рис. 1.4).

В результаті можна обчислити точну залежність між маячком (1) та власним розташуванням (2). Якщо потрібно перемістити робота в (3), можна припустити, куди і як потрібно рухатися (див. рис. 1.5).

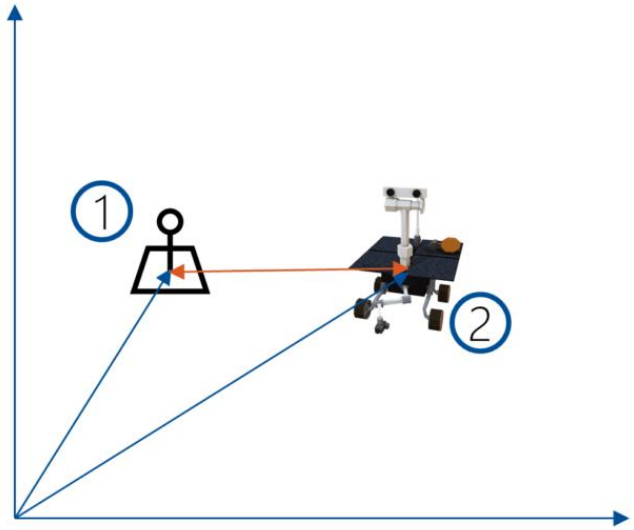


Рисунок 1.4 – Отримання інформації, щодо розташування об’єкту у просторі, (2) – робот (1) – маячок

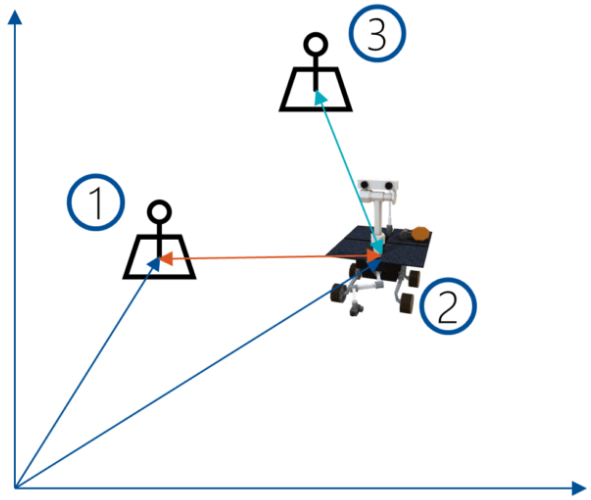


Рисунок 1.5 – Приклад руху робота (2) за наявності координат додаткової точки (3)

У реальному сценарії SLAM потрібно працювати з недосконалими знаннями, що призводить до невизначеності.

Точки мають просторові відносини одна відносно іншої. У результаті складається розподіл ймовірностей, де може бути кожна позиція. Для деяких точок може бути більш висока точність. Для інших невизначеність може бути великою. Найчастіше використовують алгоритми для обчислення позицій на основі невизначеності – це розширений фільтр Калмана, оцінка максимуму апостеріорної

імовірності (MAP – Maximum a Posteriori estimation) або корекція розшарування (Bundle Adjustment).

Через зв’язки між точками, кожне нове оновлення датчика впливає на всі позиції та оновлює всю карту. Для актуалізації всього потрібна значна кількість математики та обчислювань.

Щоб бути впевненим в надійності інформації з вимірювань доповненої реальності, узгодження нових вимірювань з попередніми знаннями є одним з найважливіших аспектів алгоритмів SLAM. Кожне вимірювання датчика містить неточності – незалежно від того, отримані вони із зображень камери або з оцінки руху кадру до кадру за допомогою акселерометрів (одометрія).

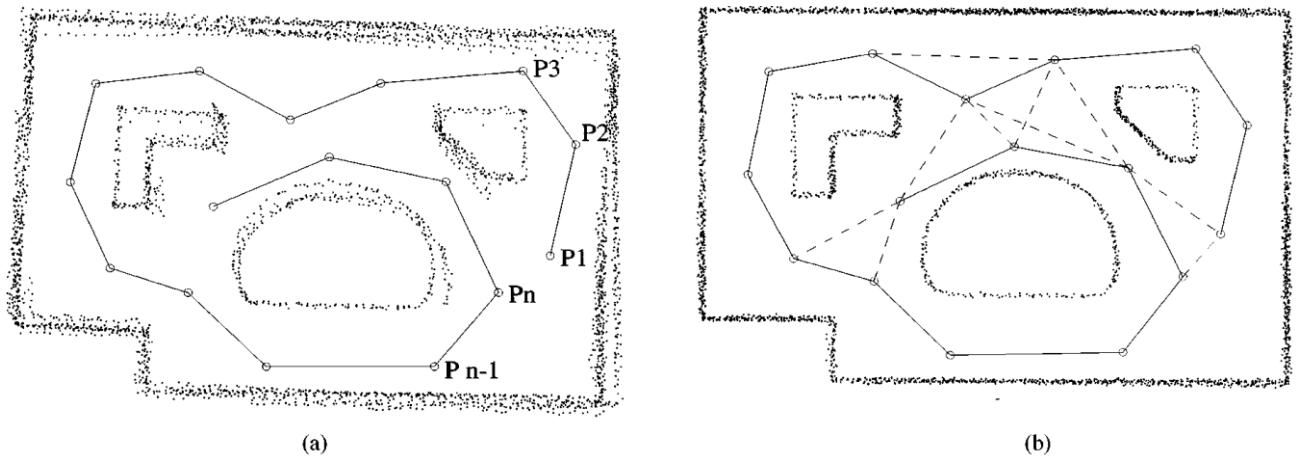


Рисунок 1.6 – (а) – відображення неточності вимірювань. (б) – Отримана картина після алгоритмів SLAM

На рис.1.6 (а) показано, як накопичуються помилки сканування діапазону з часом. Переходячи з однієї позиції P1…Pn, кожна невелика помилка вимірювання накопичується з часом, поки отримана карта середовища вже не буде точною.

Вирівнюючи скани (рис.1.6 (б)) на основі мереж відносних обмежень поз (pose), отримана відповідність значно покращується. В алгоритмі вони підтримують всі локальні кадри даних, а також мережу просторових взаємодій між ними.

Щоб доповнена реальність працювала, алгоритм SLAM повинен працювати в наступних умовах:

- Невідомий простір;
- Безконтрольна камера. Для поточної доповненої реальності на базі мобільних телефонів це, як правило, лише монокулярна камера;
- Реальний час;
- Відсутність дрейфу.

Система SLAM складається з 4 частин (див. рис. 1.7):

- Дані датчиків: на мобільних пристроях зазвичай це камера, акселерометр та гіроскоп. Це може бути доповнено іншими датчиками, такими як GPS, датчик світла, датчики глибини тощо.

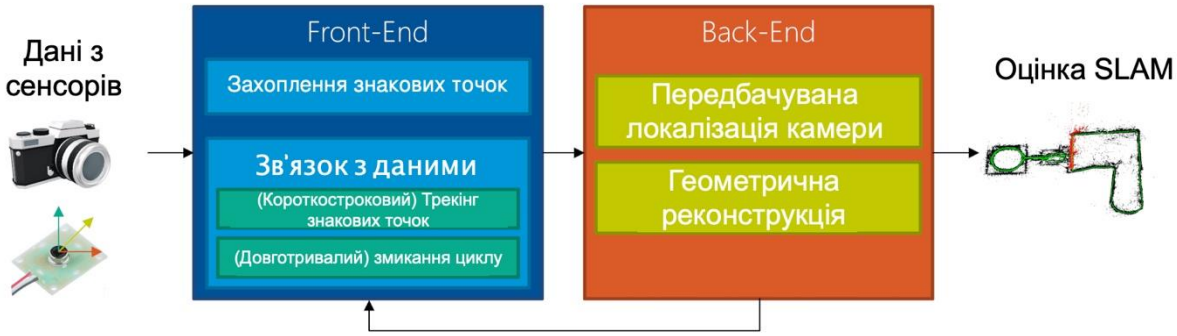


Рисунок 1.7 – Система SLAM

- Front-End: перший крок – захоплення знакових точок. Ці функції також повинні бути пов’язані з орієнтирами – ключовими точками з 3D-позицією, які також називаються точками карти. Крім того, точки відтворення карт потрібно відстежувати у відеопотоці. Довготривала асоціація (long-term association) зменшує дрейф, розпізнаючи місця, які раніше зустрічалися (змикання циклу / loop closure).
- Back-End: піклується про встановлення зв’язку між різними кадрами, локалізацію камери (pose моделі), а також відповідає за обробку загальної геометричної реконструкції. Деякі алгоритми створюють розріджену реконструкцію (на основі ключових точок). Інші намагаються зафіксувати щільну хмару 3D-точок навколишнього середовища.

- Оцінка SLAM: результат, що містить відстежувані знакові точки, їх розташування та відносини, а також положення камери у просторі.

Висновки до розділу

У першому розділі розглянуто існуючі засоби створення AR застосунків та проаналізовано програмні та апаратні засоби AR. З порівняння цих засобів видно, що кожний з них має свої переваги, недоліки та особливості. ARKit та ARCore – багатофункціональні засоби, що націлені на одну основну платформу, та вважаються технологічними лідерами у AR сфері. Інші засоби базуються на ARKit та ARCore, та додають свої особливості. Такі як, наприклад, кросплатформеність, проста інтеграція зі сторонніми засобами для розробки програмного забезпечення (Unity, VS Code).

2 ОСОБЛИВОСТІ ЗАСТОСУВАННЯ СЕРЕДОВИЩА TORCH ДЛЯ СТВОРЕННЯ ЗАСТОСУНКУ З AR

2.1 Використання сцен в Torch

Сцена Torch AR аналогічна екрану в 2D-застосунку – простір інтерфейсу для організації вмісту та взаємодії. У 2D програмі користувач має інтерактивні цифрові елементи, розташовані на екранах. У Torch AR користувач взаємодіє з інтерактивними цифровими елементами, що розташовані в сценах; ці сцени відображаються в контексті реального оточення користувача.

Сцена містить набір об’єктів, взаємодії об’єктів та положення об’єктів відносно опорної точки (World anchor). Усі сцени в рамках проекту та всі об’єкти в межах цих сцен мають спільну опорну точку.

В рамках проекту можна додати нові сцени, змінити назву або опис сцени, переглянути список усіх сцен, копіювати сцени, та видалити сцени. Диспетчер сцени, розташований у верхньому центрі екрана, відображає назву сцени, в якій зараз знаходиться застосунок.

Сцена концептуально відрізняється від реального середовища, в якому цю сцену можна пережити. Зв’язок між реальним світом та сценою – опорна точка (World anchor). За замовчуванням орієнтація опорної точки така ж, як і напрямок, в якому спрямована камера (рис. 2.1).

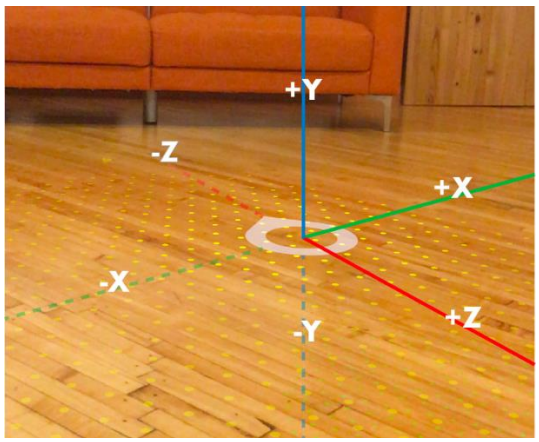


Рисунок – 2.1 Представлення опорної точки (World anchor) в Torch

Опорна точка є орієнтиром для всього проекту: кожен об’єкт у кожній сцені проекту розміщується відносно опорної точки [10]. Цей крок необхідний для більшості застосунків AR, які використовують ARKit (Apple) або ARCore (Google).

2.2 Експорт проекту в архів .torchkitproj

Torch проекти можуть бути стиснуті та експортовані у архів формату torchkitproj. В цьому архіві міститься інформація про моделі, їх положення відносно опорної точки, та JSON (Java Script Object Notation) файл з метаданими. На рисунку 2.2 в архіві містяться папки з ресурсами (картинки, моделі в форматі USDZ), та файл project.json з інформацією про сцени та розташування об’єктів в проекті.

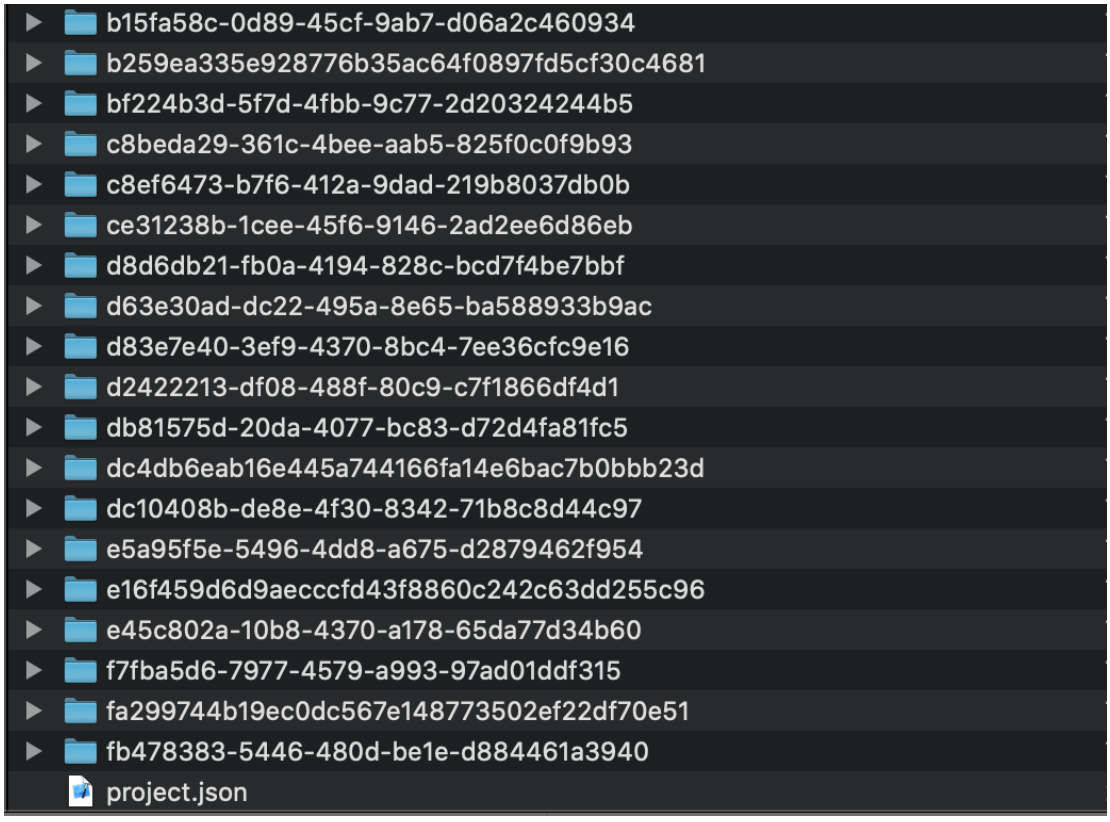


Рисунок 2.2 – Вміст архіву torchkitproj

Моделі зберігаються в форматі USDZ. Формат файлу USDZ був створений партнерством Apple та Pixar; це було оголошено на Всесвітній конференції

розробників Apple, навесні 2018 року [11]. USDZ будується поверх існуючого формату USD (інший формат, створений для зберігання 3D-моделей та анімацій), але додає застосункові оптимізації та сумісність для використання з мобільними пристроями.

USD пропонує багато функцій для зберігання 3D-вмісту, включаючи:

- Надійні схеми взаємодії геометрії, шейдингу та деформації скелета моделі
- Висока продуктивність пошуку та надання даних, включаючи потужні функції копіювання
- Можливість нативно компресувати вміст, що обирається користувачем
- Звукова архітектура, яка досить гнучка, щоб адаптуватись до майбутніх потреб

USDZ - це zip архів файлу USD з нульовим стисненням. Пакет USDZ може містити лише типи файлів, дані яких можуть використовуватися під час виконання файлу USD:

- usda, usdc, usd файли
- Файли png та jpeg (будь-яке з декількох поширених розширень для jpeg) для зображень / текстур
- Файли M4A, MP3, WAV для вбудованого аудіо (надаються в порядку бажаного формату)


```
{
   "name":"NAMU",
   "scenes":[ ],
   "exportId":"88124b08-3c2f-4bda-876b-c566d84f3f5a",
   "version":12,
   "ownerDisplayName":"Danil Kurilo",
   "createdAt":"2019-09-20T10:20:52.944Z",
   "assets":[ ]
}
```


Рисунок 2.3 – структура JSON файлу

В корні JSON файлу, що зберігається в архіві, містяться дані про проект, масив з сценами та масив з ресурсами проекту.

В масиві assets (рис.2.4) зберігається інформація про ресурси проекту:

- mainFile – назва файлу в папці архіву;
- displayName – назва файлу в проекті Torch;
- assetId – назва папки в архіві USDZ, де зберігається цей файл;
- thumbnailUrl – посилання на скомпресований файл для відображення в проекті Torch на попередньому перегляді файлу;
- type – тип файлу.


```
"assets":[
  {
    "mainFile":"image.png",
    "displayName":"btn DALI.png",
    "assetId":"c8ef6473-b7f6-412a-9dad-219b8037db0b",
    "revisionId":"44a6d578-536f-4ee9-9d7d-812efeb1ad74",
    "buildId":"b45f2fd7-a968-4f92-b452-5eea06beb5dd",
    "buildName":"c8ef6473-b7f6-412a-9dad-219b8037db0b$44a6d578-536f-4ee9-9d7d-812efeb1ad74",
    "thumbnailUrl":"https://storage.googleapis.com/torchbase-godot-assets/c8ef6473-b7f6-412a-9dad-219b8037db0b%2444a6d578-536f-4ee9-9d7d-
812efeb1ad74%2F1570183315%2Fthumbnail.png?GoogleAccessId=torchbase-
124e0%40appspot.gserviceaccount.com&Expires=1571924046&Signature=P4AYq7pHw%2FdTw7LYGfZzNGf%2Bp8xRNOZ1fGVLNvxrCMXlSzJ%2BOfX4lr5rpxCCTosAnCScNCO3pscUtM5xrMjt
    "type":"image",
    "metadata":"CAEQxgUY7gE="
  },
```


Рисунок 2.4 – Структура об’єкту asset

У об’єкті scene (рис. 2.5) міститься інформація про сцену та її тригери:

- position – відстань у метрах від опорної точки;
- description – текстовий опис сцени;
- state – початковий стан сцени (містить інформацію про подальші зміни в сцені);
- trigger – умова, за якої виконується triggerResponse;
- triggerResponse – набор команд, що повинні виконатись про виконанні умов trigger;
- triggerType – тип тригеру (об’єкт попав у камеру, на об’єкт натиснули, об’єкт попав у радіус, тощо).

```
"scenes":[
   {
      "position":0,
      "description":"used to set the anchor",
      "state":{
         "00000000-0000-0000-0000-000000000000":{
            "trigger_response":{ },
            "id":"00000000-0000-0000-0000-000000000000",
            "trigger":{
               "8fddaaee-ea14-4e8b-b837-3fbf8b06d9c1":{
                  "id":"8fddaaee-ea14-4e8b-b837-3fbf8b06d9c1",
                  "displayName":"hall image found",
                  "data":{
                     "assetId":"ec1ee173-f640-4468-b84c-c36fbfe19389",
                     "infiniteTrigger":true,
                     "finiteTriggers":0,
                     "distanceMeters":0.1
                  },
                  "userNamed":true,
                  "type":"found_image"
               }
            },
            "components":{ },
            "type":"scene",
            "children":[ ]
         },
         "ec1ee173-f640-4468-b84c-c36fbfe19389":{
            "orientation":"orientation_up",
            "assetId":"3485d4c5-5400-4912-9597-e5991bf2c733",
            "widthMeters":0.5,
            "editOrientation":1,
            "type":"trackedImage",
            "id":"ec1ee173-f640-4468-b84c-c36fbfe19389"
         }
      },
      "id":"1Eqgf6qPrpAcvQYAMe54",
      "name":"initial"
   },
```


Рисунок 2.5 – Структура об’єкту scene

2.3. Інтеграція SDK через CocoaPods

CocoaPods – це інструмент управління залежностями бібліотек для застосунків OS X та iOS [12]. За допомогою CocoaPods можна визначати залежності

проекту, які називаються pods, і легко керувати їх версіями протягом часу та в різних середовищах розробки. CocoaPods написано на Ruby.

CocoaPod/pod – це компонент, орієнтований на користувача, і активується щоразу, коли викликається команда pod. Він включає в себе всю функціональність, необхідну для фактичного використання CocoaPods. Podfile – це файл, який визначає pod-и, які будуть використовуватись у проекті.

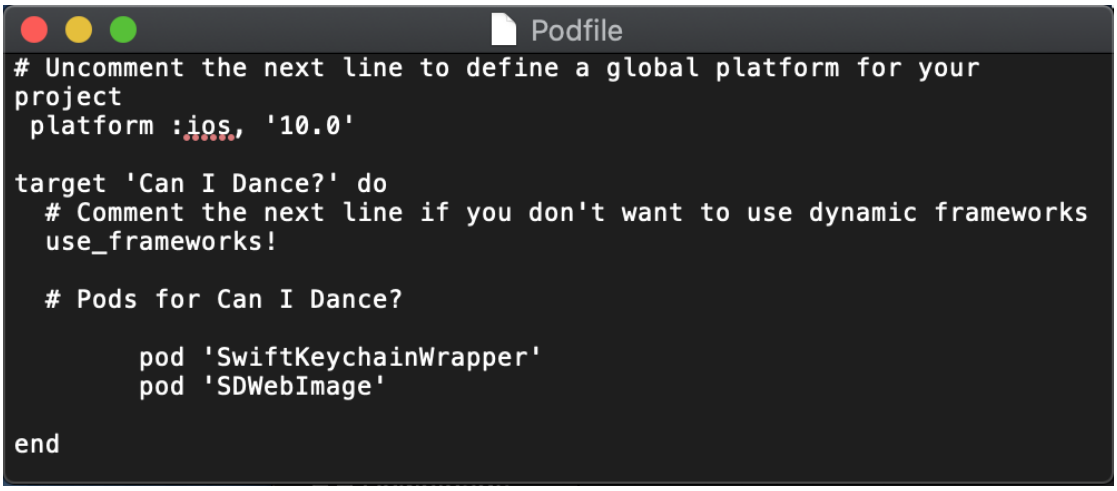


Рисунок 2.6 – podfile з налаштуванням залежностей

.Podspec – це файл, який визначає, як певний pod додається до проекту. Він підтримує такі функції, як перелік вихідних файлів, фреймворки, прапорці компілятора та будь-які інші залежності, необхідні бібліотеці.

Для того, щоб додати Torch SDK до Xcode проекту у podfile потрібно виконати команду (рис. 2.7):

```
pod 'TorchKit', :http => "https://artifacts.torch.dev/torchkit/TorchKit_0_3_2.zip"
```

Рисунок 2.7 – Імпорт TorchKit в проект

При виконанні pod install, TorchKit буде завантажено та додано в проект.

Висновки до розділу

У другому розділі детально розглянуто бібліотеку Torch SDK, зокрема використання сцен, як окремих AR-сцен та концепцію опорних точок. За допомогою сцен, при правильному виборі сценарію застосунку можна побудувати оптимальну та багатофункціональну архітектуру, що дозволить легко завантажувати та видаляти сцени з пам’яті пристрою без тривалих пауз.

Виявлено, що архів torchkitproj можна використовувати як базу даних з інформацією про переходи між сценами, а також даними про наявність, кількість тригерів та їх результати.

Встановлено, що Torch SDK має єдиний спосіб інтеграції у застосунки – через менеджер залежностей CocoaPods. Зазанчений спосіб підтримує динамічні фреймворки.

3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОЕКТУ TORCH ДЛЯ ІМПОРТУВАННЯ В ЗАСТОСУНОК

Головним завданням у процесі створення застосунку з доповненою реальністю для Національного художнього музею України було створити інструмент, завдяки якому користувач буде мати можливість дізнатись додаткову інформацію про твори, що представлені у музеї з підвищеною зацікавленістю.

Додатковим завданням було надати можливість співробітникам музею змінювати чи доповнювати інформацію, що міститься в застосунку без необхідності писати програмний код, чи оновлювати застосунок.

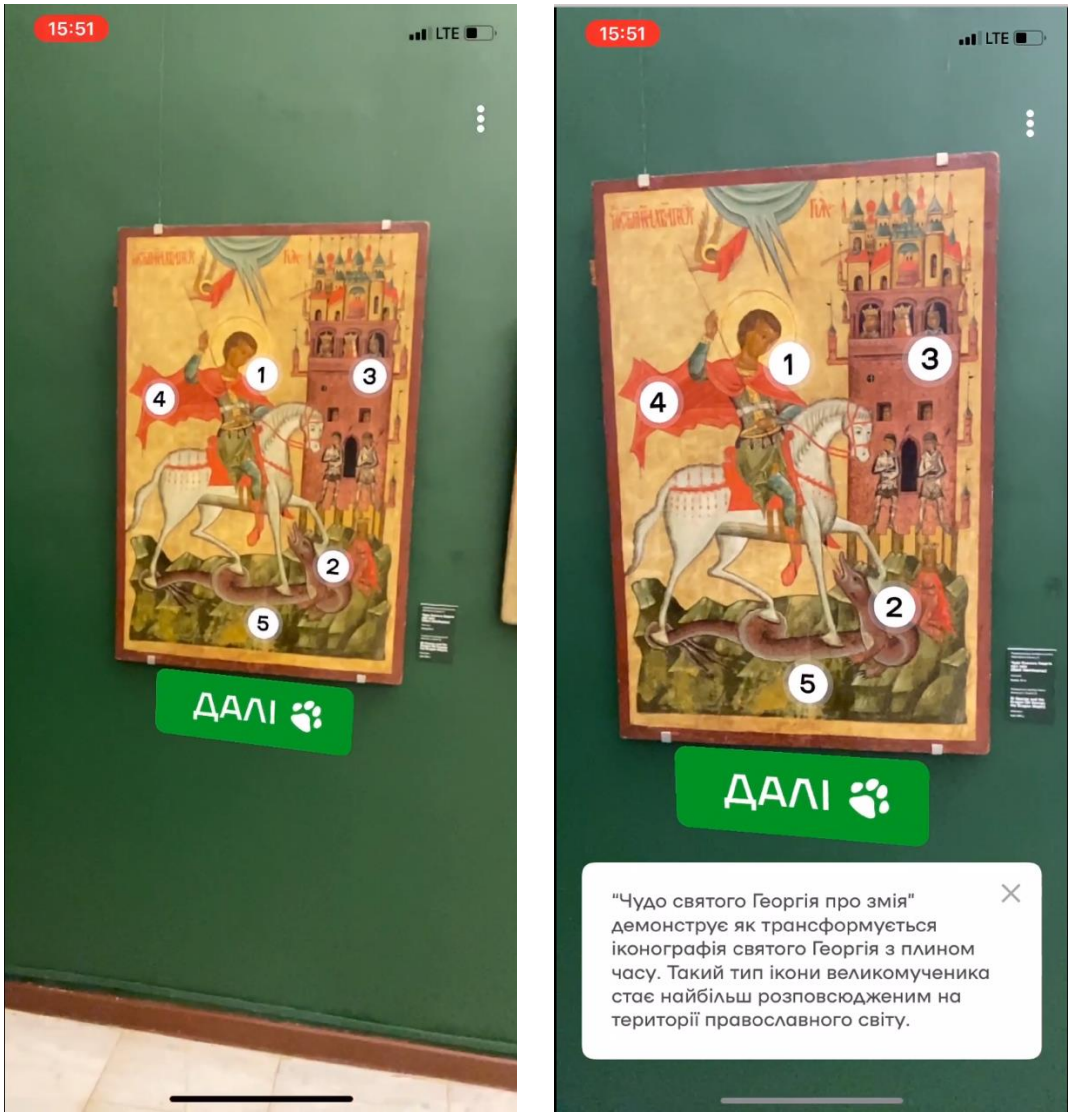
Для вирішення поставленої задачі обрано бібліотеку Torch SDK, мову програмування iOS та фреймворк для створення доповненої реальності ARKit.

3.1 Сценарій роботи застосунку в AR

Основна ідея застосунку полягає в тому, щоб спрямовувати користувача від однієї картини до наступної, та при взаємодії з картиною – показувати ключові точки, при натисканні на які відкриваються вікна з текстовим описом. Навігація від картини до картини відбувається за допомогою розміщення 3D моделей «лапок» відносно опорних точок поточної сцени (рис. 3.1).

Рисунок 3.1 – Навігація до картини

При попаданні в кадр шуканої картини та вдалому трекінгу – сцена змінюється на наступну, та відображаються ключові точки (рис. рис. 3.2 а)). Натискання на одну з ключових точок відкриває текстовий опис цієї точки (рис. рис. 3.2 б)).



а) б)

Рисунок 3.2 – а) відображення ключових точок на картині, б) опис ключової точки

Натискання на кнопку «Далі» відкриває наступну сцену, яка веде користувача до наступної картини.

Такий цикл повторюється для кожної картини у застосунку. Єдиний етап, що відрізняється – презентація 3D моделі церкви (рис. 3.3).

Рисунок 3.3 – 3D модель церкви

На рисунку 3.4 зображена діаграма роботи застосунку.

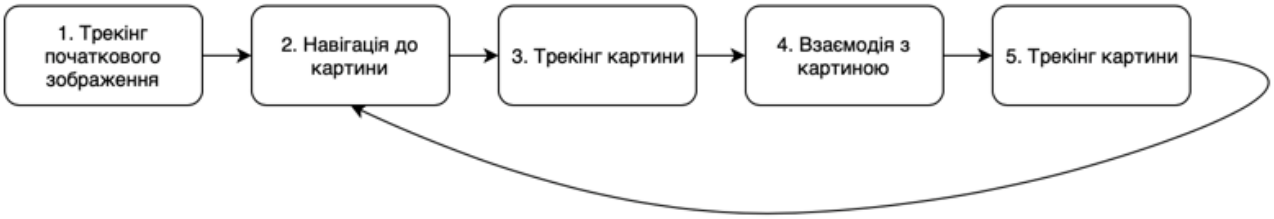


Рисунок 3.4 – Діаграма роботи застосунку

Перший етап – трекінг початкового зображення. Для того, щоб не змушувати користувача сканувати поверхні та виставляти опорну точку власноруч, потрібно створити додаткову порожню сцену в застосунку, що буде вмикати наступну сцену, коли застосунок захопить камерою початкове зображення.

У першої сцени не буде опорної точки. Щоб досягти цього треба ініціалізувати першу сцену без пошуку опорної точки таким чином:

```
if self.projectAnchorManager == nil {
  self.projectAnchorManager = ProjectAnchorManager(withSceneView: self.sceneView) { [weak self] in
    guard let projectVC = self, let torchProj = projectVC.torchProject else { return }
```

Рисунок 3.5 – Налаштування ініціалізації сцени без пошуку опорної точки

Файл, що виконує описану команду представлено у додатку Б (див Б.7)

У першій сцені потрібно додати тригер Image Found -> Switch Scene, що буде перемикати сцену на наступну при знаходженні зображення (рис.3.6). У другій сцені опорною точкою буде саме це зображення

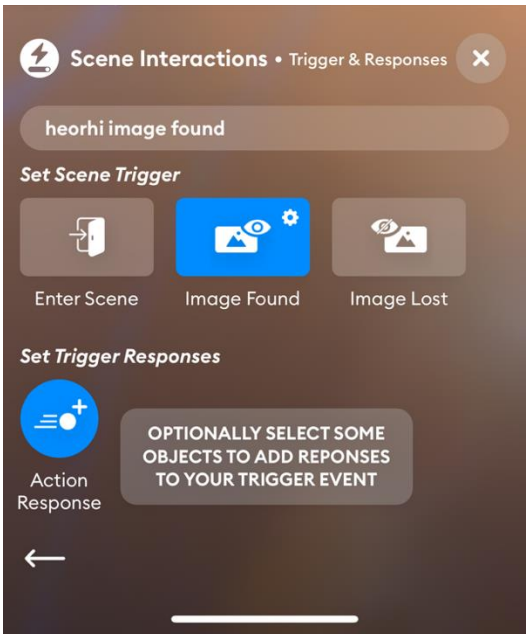


Рисунок 3.6 – налаштування тригеру Image Found -> Switch Scene

Таким чином, користувачу потрібно не виставляти ключову точку, а просто знайти початкову картинку. В даному випадку це табличка «Початок експозиції» в холі музею (рис.3.7).

Рисунок 3.7 – табличка, що є початковою картиною

Другий етап – навігація до картини. Відштовхуючись від опорної точки, потрібно виставити 3D моделі, що будуть вести користувача до наступної картини. Для збільшення ефекту навігації, можна виставити на кожну модель по 2 тригери – Enter proximity та Exit proximity. Ці тригери спрацьовують коли користувач, відповідно, заходить та виходить з радіусу об’єкту (в метрах). Trigger response в даному випадку потрібно виставити visibility (видимість) моделі.

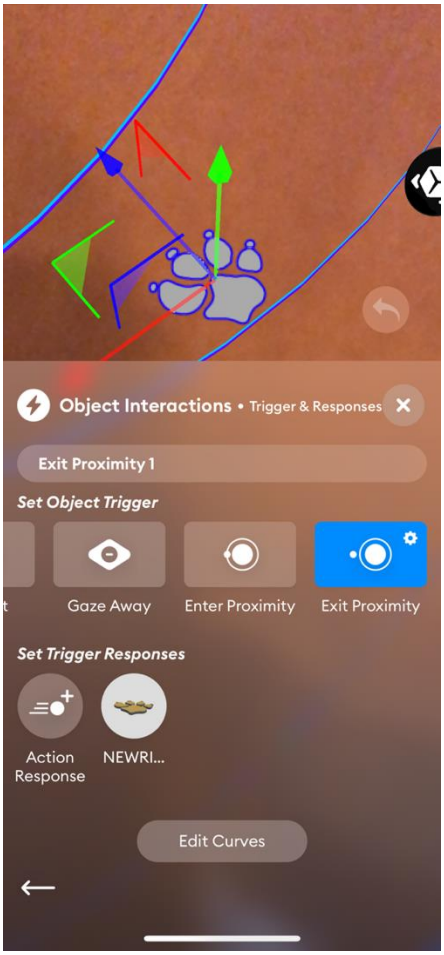


Рисунок 3.8 – Налаштування тригеру входу/виходу з радіусу об’єкту

Таким чином, коли користувач буде в достатній близькості від моделі – вона з’явиться. Це вирішує проблему появи моделей за стінами, коли користувач не повинен їх бачити.

Також, в сцені з навігацією потрібно налаштувати tracked image – зображення, яке спровокує запрограмовану дію (рис. 3.9).

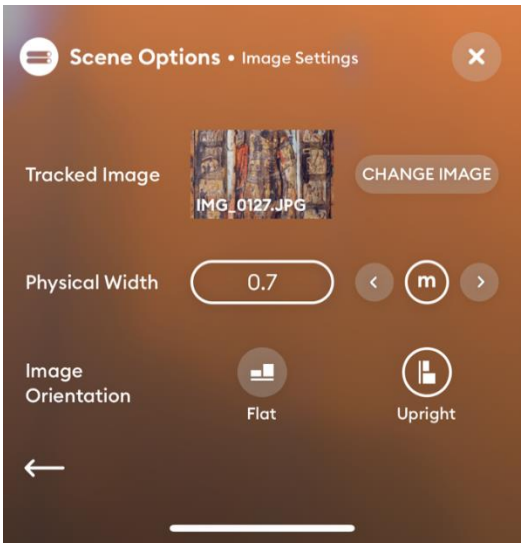


Рисунок 3.9 – Налаштування tracked image

Trigger response на цьому зображенні буде перемикання сцени.

Третій етап – трекінг картини. Коли сцену буде переключено за допомогою знайденої картини, у наступній сцені опорною точкою буде така ж сама картина (рис. 3.10). Але зменшуючи відстань користувача від опорної точки, програма зменшує можливість та величину похибки.

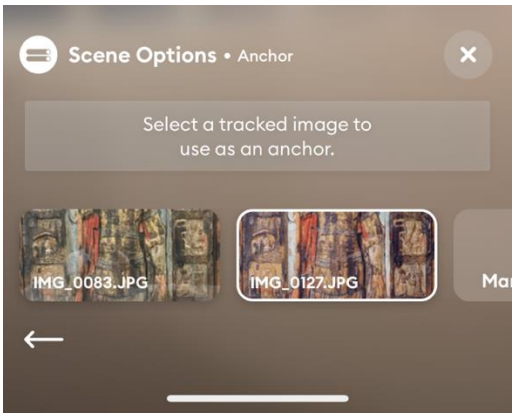


Рисунок 3.10 – Налаштування картини як опорної точки сцени

Четвертий етап – взаємодія з картиною (рис. 3.11). Потрібно додати моделі кнопок в форматі png чи jpg, та розставити їх по картині. На кожну з цих кнопок налаштувати trigger – button tapped, що буде спрацьовувати коли на модель натискають.

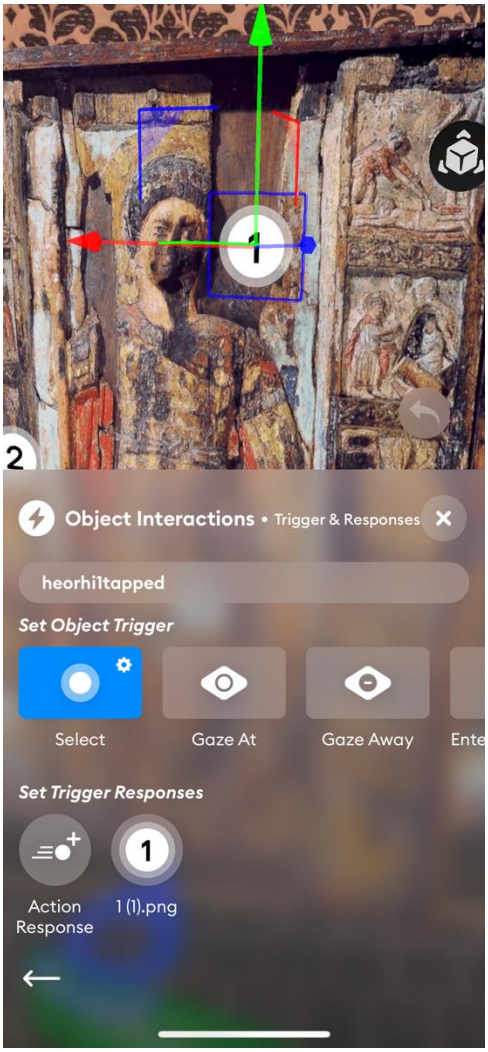


Рисунок 3.11 – Налаштування тригеру на кнопці на картині

П’ятий етап – завершення інтеракції (взаємодії) з картиною. По натисканню на кнопку «Далі» сцену буде переключено на наступну. Опорною точкою залишається та ж сама картина, але логіка в сцені аналогічна із логікою в другому етапі. Тепер 3D моделі для навігації повинні бути виставлені відштовхуючись від попередньої картини.

3.2. Налаштування ARKit під Torch SDK

Для використання Torch SDK у застосунку, потрібно отримати API ключ на веб сайті Torch. Цей ключ потрібен для того, щоб тільки зарегістровані розробники могли відправляти на отримувати інформацію з серверів Torch. Після отримання ключа потрібно при кожному запуску застосунку перевіряти цей ключ на дійсність.

```
do {
    try TorchKit.shared.initSDK(apiKey:
        "djIucHVibGljLmV5SndjbTkwYnlJNklrTkJSVk5JUjNCTlVURnJlRmRZYkVaT2JXUlRVVlJrUlZGV2FFMVNSM1JRVlZSSmVtUnJWa3ROVkVsaFJFRnFjaTlrY25KQ1VrTkJ
        NRzlITlVGVFNWTkRhRUpDUm1wTWQyMXVlVVZ5U1Vwa1F6WkdZbEpQVjIwaWZjVHVrTXV5cmZHY2VBS21QQVFEajdmdVlYUVhQaFgtTUlpbzR4OWRwV1gtYTlKdHVjaVRJRUZ
        FLW5ORlhnb2JhYzdPVm1vV2Zra0Eyc2d3MlVvTVBBQQ==")
} catch {
    os_log(OSLogType.default, "Init failed!")
}
```

Рисунок 3.12 – перевірка API ключа при запуску застосунку

Файл, що виконує описану команду представлено у додатку Б (див Б.1)

Контролер, що буде відповідати за AR сцену, повинен бути делегатом (передавати дані по визову однієї з функцій іншого контролеру) ARSCNViewDelegate та ARSessionDelegate

```
class TorchProjectViewController: UIViewController, ARSCNViewDelegate, ARSessionDelegate {
```

Рисунок 3.13 – Наслідування класу TorchProjectViewController

При запуску застосунку потрібно задати параметри AR сесії, з якою працює Torch SDK.

```
93          let configuration = ARWorldTrackingConfiguration()
94          //PLANE DETECTION
95          configuration.planeDetection = [.horizontal]
96          self.sceneView.session.run(configuration)
97          self.sceneView.session.delegate = self
98          UIApplication.shared.isIdleTimerDisabled = true
```

Рисунок 3.14 – налаштування AR сесії

На рядку 93 створюється нова константа, що буде тримати налаштування AR сесії. На рядку 95 – обирається тільки трекінг горизонтальних поверхонь. На рядку

96 запускається AR сесія (sceneView) з обраними налаштуваннями (configuration). На рядку 97 даний контролер становиться делегатом цієї AR сесії.

3.3. Нативний UI поверх TorchKit

Для створення та відображення нативного UI у застосунку, що використовує Torch потрібно працювати з інтерфейсом програмно, не використовуючи storyboard-и.

Сторіборд (storyboard) – це візуальне зображення користувальницького інтерфейсу програми iOS, що показує екрани вмісту та зв'язки між цими екранами. Сторіборд складається з послідовності сцен, кожна з яких представляє контролер та його view; сцени з'єднані зв'язками (segue), що представляють собою перехід між двома контролерами.

Xcode надає візуальний редактор для сторібордів, де викладається та проектується користувальницький інтерфейс програми, шляхом додавання на сцени вмісту, такого як кнопки, таблиць та тексту.

З інтерфейсу в застосунку є кнопка «меню», меню з підпунктами у вигляді таблиці, та вспливаючі вікна, що появляються при натиску на одну з кнопок на картині.

Інтерфейс, що конфліктує з AR-сценою:

- Кнопка «Меню»:

Програмно кнопка створюється таким чином, як показано на рис. 3.15.

```
@IBOutlet weak var menuButton: UIButton!
```

Рисунок 3.15 – Створення кнопки програмно

Для того, щоб кнопку було видно, AR сцену потрібно розміщати за кнопкою (рис. 3.16).

```
self.view.insertSubview(self.sceneView, belowSubview: menuButton)
```

Рисунок 3.16 – Розміщення AR сцени під кнопкою

Екран «Меню» створений за допомогою сторіборду буде виглядати, як показано на рис. 3.17.

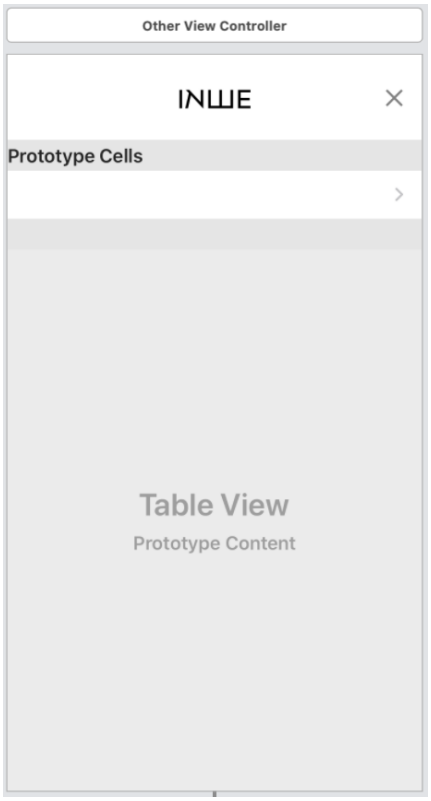


Рисунок 3.17 – Екран «меню» у сторіборді

Для презентації меню цей код спрацьовує при натисканні на кнопку «меню»

```
@IBAction func menuButtonPressed(_ sender: UIButton) {
    let menuVC = storyboard?.instantiateViewController(withIdentifier: "menuVC") as! OtherViewController
    present(menuVC, animated: true, completion: nil)
    menuVC.torchProjDelegate = self
}
```

Рисунок 3.18 – Код презентування екрану «Меню»

Вспливаючі вікна, що з’являються при взаємодії з картинами (рис 3.19):

```
func presentPopUp(fromBottom: Bool) {
    var attributes = EKAttributes()

    attributes.position = .bottom
    attributes.entryBackground = .color(color: .white)
    attributes.displayDuration = .infinity
    attributes.entryInteraction = .absorbTouches
    attributes.screenInteraction = .forward
    attributes.positionConstraints.verticalOffset = 10
    attributes.roundCorners = .all(radius: 10)
    let widthConstraint = EKAttributes.PositionConstraints.Edge.ratio(value: 0.9)
    let heightConstraint = EKAttributes.PositionConstraints.Edge.intrinsic
    attributes.positionConstraints.size = .init(width: widthConstraint, height: heightConstraint)

    if !fromBottom {
        attributes.position = .top
    }

    SwiftEntryKit.display(entry: popUpView, using: attributes)

}
```

Рисунок 3.19 – Функція, що презентує спливаюче вікно

Файл, що виконує описані команди представлено у додатку Б (див Б.7)

Для презентації спливаючих вікон було використано бібліотеку SwiftEntryKit.

У реалізації функції що презентує вспливаючі вікна потрібно визначити змінну, що буде відповідати за зовнішній вигляд вікон – EKAttributes(). Функція приймає одне значення fromBottom. За допомогою цього можна обирати положення вікна (зверху/знизу). Після визначення вигляду вікна воно представляється за допомогою SwiftEntryKit.display.

3.4 Callback функції в TorchKit

Callback функції – це автономні блоки функціональності, які можна передавати і використовувати у коді. Callback в Swift схожі з блоками на C і Objective-C та лямбдами в інших мовах програмування [13].

Callback функції можуть захоплювати та зберігати посилання на будь-які константи та змінні з контексту, в якому вони визначені. Це відомо як закриття цих констант і змінних.

Callback функції в Torch існують в спеціальному класі – TorchProjectNode. TorchProjectNode - клас, що завантажує .torchkitproj та генерує дочірні вузли для виконання проекту Torch. Він обробляє завантаження та вивантаження ресурсів, створюючи дочірні вузли та реагуючи на оновлення в сцені. Однак є кілька функцій підтримки, які потрібно викликати, щоб все працювало:

Функцію tick потрібно викликати кожен кадр для просування стану проекту. Функція tap повинна бути викликана, коли об’єкт був натиснутий.

В TorchProjectNode існують такі callback функції:

- sceneChanged;

Якщо налаштований, цей callback буде викликано, коли сцена зміниться в проекті Torch.

public var sceneChanged: ((TorchKitSceneIdName) -> Void)?

- triggerFired.

Якщо налаштований, цей callback буде викликано коли в проекті спрацьовує тригер.

public var triggerFired: ((TorchKitSceneObjectIdName) -> Void)?

- triggerFinished.

Якщо налаштований, цей callback буде викликано коли всі відповіді на тригер (trigger Response) закінчені.

- tick.

tick продовжує виконання проекту. Його слід викликати з методу *SCNSceneRendererDelegate renderer (_:updateAtTime :).*

public func tick(delta: Double, cameraTransform: simd_float4x4, currentGazedNode: SCNNode?)

де:

- *delta* – Зміна часу від останнього виклику tick. Одиниці вимірювання - секунди.

- *cameraTransform* – Поточне положення камери. Може *передаватися безпосередньо з arSession.currentFrame.camera.transform*, якщо використовується ARKit.
- *currentGazedNode* – SCNNode, на який зараз вказує камера. Використовується метод .hitTest на SCNScene, щоб знайти цей node.
- tap.

tap викликається, коли користувач натиснув SCNNode. Це обробляється автоматично, якщо використовується *TorchGestureManager*.

public func tap(node: SCNNode)

В проекті callback функції використовуються для відображення спливаючих вікон. На кожній кнопці на картині знаходиться тригер з відповідною назвою. Коли користувач натискає на одну з кнопок, спрацьовує callback функція triggerFired. У проекті потрібно відслідковувати ці моменти, та реагувати обробляючи дані, що прийшли з відповіддю на callback функцію.

На рисунку 3.20 зображено реагуючу функцію, де torchProject – посилання на проект Torch, triggerFired – callback функція, trigger – посилання на тригер, що запустив callback функцію.

```
torchProject?.triggerFired = { trigger in
```

Рисунок 3.20 – Функція, що реагує на callback функцію triggerFired

На рисунку 3.21: trigger.id – унікальна програмна назва тригеру, trigger.name – назва тригеру, що визначена у проекті Torch, trigger.sceneId – ідентифікатор сцени, в якій спрацював тригер.

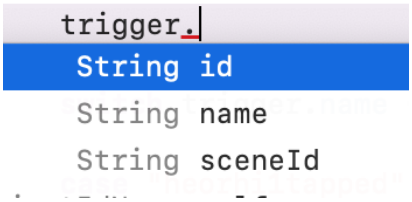


Рисунок 3.21– Інформація, що міститься в посиланні на тригер

На рисунку 3.22 зображено перебирання варіантів значення trigger.name.

```
switch trigger.name {

case "heorhi1tapped":
    self.popUpDescription.text = artObjects[0].trigger1description
    self.presentPopUp(fromBottom: true)
case "heorhi2tapped":
    self.popUpDescription.text = artObjects[0].trigger2description
    self.presentPopUp(fromBottom: true)
case "heorhi3tapped":
    self.popUpDescription.text = artObjects[0].trigger3description
    self.presentPopUp(fromBottom: true)
case "heorhi4tapped":
    self.popUpDescription.text = artObjects[0].trigger4description
    self.presentPopUp(fromBottom: true)
case "heorhiNextTapped":
    self.popUpDescription.text = artObjects[0].nextconnection
    self.presentPopUp(fromBottom: false)
```

Рисунок 3.22 – Приклад обробки результатів callback функції triggerFired

Файл, що виконує описану команду представлено у додатку Б (див Б.8)

На рисунку 3.23 показано, як застосування перелічених функцій дозволить керувати спливаючими вікнами.

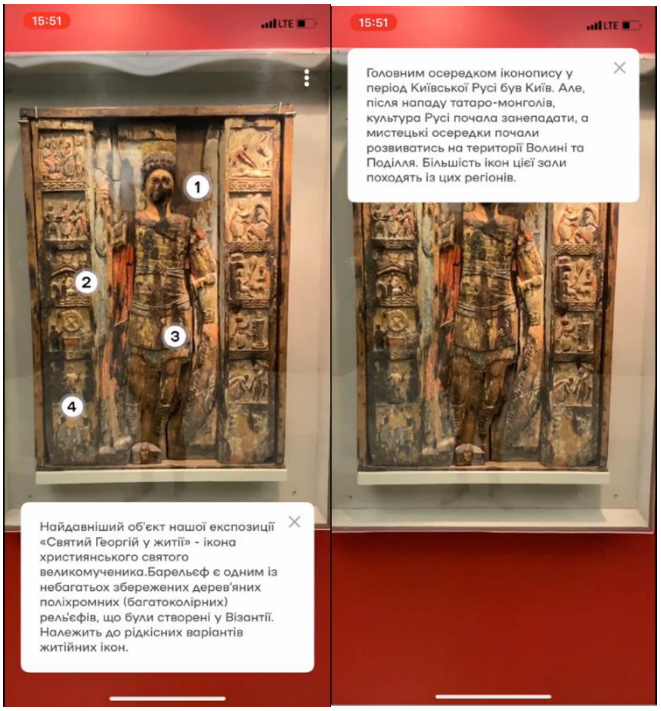


Рисунок 3.23 – Презентація спливаючих вікон

Якщо на кнопці, яку натиснув користувач спрацював тригер з назвою “heorhi1tapped”, текст у спливаючому вікні змінюється на опис першої ключової точки картини «Святий Георгій у житії» та викликається функція *presentPopUp*, що презентує спливаюче вікно.

У випадку, коли натиснута кнопка «Далі» (*case “heorhiNextTapped”*) ця функція викликається з параметром *fromBottom = false*, тобто вікно буде представлено зверху.

3.5. Використання Google Drive як серверу проекту

Диск Google (*Google Drive*) – це сховище даних, яке належить компанії Google, що дозволяє користувачам зберігати свої дані на серверах у хмарі та ділитися ними з іншими користувачами в Інтернеті [14]. Google Drive включає Google Docs, Sheets, and Slides, офісний пакет, який дозволяє спільно редагувати документи, електронні таблиці, презентації, малюнки, форми, і багато іншого.

Для проекту було створено таблицю в Google Sheets з даними про картини.

A	B	C	D	E	F	G	H	I	J	K
Id	Name	Authoryear	Image	Trigger1Name	Trigger1Description	Trigger2Name	Trigger2Description	Trigger3Name	Trigger3Description	Trigger4Name
1	Святий Георгій у житії	Візантія XII ст	https://drive.google.com/uc?id=17	heorhi1tapped	Найдавніший об'єкт	heorhi2tapped	Георгій був воєначал	heorhi3tapped	Історія ікони така ж	heorhi4tapped
2	Чудо святого Георгія про змія	Друга половина XV ст	https://drive.google.com/uc?id=1H	horse1tapped	"Чудо святого Георгія	horse2tapped	На іконі зображено а	horse3tapped	Готичні шпилі свідча	horse4tapped
3	Великомучениці Варвара і Катерина	1740-ві роки. Північне Лівобережжя	https://drive.google.com/uc?id=15	velyk1tapped	У 17-му столітті на т	velyk2tapped	На парній іконі зобра	velyk3tapped	Християнство - реліг	nil
4	Портрет Данила Єфремовича	Київ, 1762	https://drive.google.com/uc?id=16	ded1tapped	Парадно-репрезента	ded2tapped	На портреті зображе	ded3tapped	Як і на іконах, предм	nil
5	Козака з ляхом разговор	Центральна Україна, XIX ст	https://drive.google.com/uc?id=1M	mamay1tapped	Існує декілька версій	mamay2tapped	Запорізького війська	mamay3tapped	Спочатку в сюжеті к	mamay4tapped
6	Портрет літньої селянки	Опнас Рокачевський, 1860	https://drive.google.com/uc?id=17	babushka1tapped	У першій половині 19	babushka2tapped	Художник Опанас Ро	nil	nil	nil
7	П'яного везуть	Костянтин Трутовський, 1861	https://drive.google.com/uc?id=1kt	drunk1tapped	Жанровий живопис г	drunk2tapped	Уся творчість Костян	nil	nil	nil
8	Рання весна	Кіріак Костанді, 1890	https://drive.google.com/uc?id=1a	monk1tapped	У другій половині 19	monk2tapped	Кіріак Костанді - оди	nil	nil	nil
9	Ідилія	Микола Пимоненко, 1908	https://drive.google.com/uc?id=1l2	love1tapped	У 1875 році О. Мура	love2tapped	Микола Пимоненко у	nil	nil	nil
10	Затишшя	Володимир Орловський, 1890	https://drive.google.com/uc?id=1Z	forest1tapped	Як ми вже розглянул	forest2tapped	Володимира Орловсь У ранній творчості О У своїй творчості ми	nil	nil	nil
11	Село взимку	Петро Левченко, початок 1900х	https://drive.google.com/uc?id=1G	winter1tapped	На межі XIX – XX ст	winter2tapped	Петро Левченко - вид Зобразити сніг - у всі	nil	nil	nil

Рисунок 3.24 – Структура даних в google sheets

Назви колонок на рисунку 3.24:

- id – порядковий номер картини;
- Name – назва картини;
- Authoryear – автор картини, дата написання;
- Image – посилання на файл зображення картини, що зберігається на Google Drive;
- Trigger_X_Name – назва тригеру;

- Trigger_X_Description – текст, що відображається у спливаючому вікні коли спрацьовує відповідний тригер.

Для обробки цього документу було використано сервіс http://gsx2json.com. Цей API підключається до електронної таблиці та очищає дані, забезпечуючи простий читабельний JSON, який приймає http запити [15]. Потім цей JSON доступний за допомогою кінцевої точки api:

http://gsx2json.com/api?id=SPREADSHEET_ID&sheet=SHEET_NUMBER&q=QUERY

де:

- SPREADSHEET_ID - Ідентифікатор документа. Великий довгий буквено-цифровий код посередині URL-адреси документа
- SHEET_NUMBER - Номер індивідуального аркуша, з якого отримуються дані. Якщо аркуш не введений, то за замовчуванням = 1.
- QUERY – застосункові параметри запиту:
- Integers – якщо «true», чисельні значення будуть приходити у вигляді текстових даних.
- Rows – якщо «false», запит повертає інформацію тільки про колонки в таблиці
- Columns – якщо «false», запит повертає інформацію тільки про рядки в таблиці

На рисунку 3 25 у якості демонстрації показано скрін частини відповіді на запит до сервісу у форматі JSON.

Рисунок 3.25 – Частина відповіді на запит до сервісу у форматі JSON

Як видно з рисунку 3.25, «сирий» JSON масив дуже великий та не структурований. Тому у застосунку проводиться парсинг JSON даних (рис.3.26). Парсинг – структуризація та впорядкування великого об'єму даних.

```
var artObjects = [ArtObject]()

class PopUpContentManager {

    private init() {}
    static let shared = PopUpContentManager()

    //var artObjects = [ArtObject]()

    func loadData() {

        guard let url = URL(string:
            "http://gsx2json.com/api?id=1f7j3WPBouLWuIhySO2p2bFF13RlR76BxvLSNFUT9Ods&columns=false") else {
            print("RETURNED HERE?????")
            return }
        //http://gsx2json.com/api?id=11tvOkkWUcaYtmk5nVNOzMNx4v19VnhYuIqjC60zVrSw&sheet=1&columns=false
        URLSession.shared.dataTask(with: url) { (data, response
                , error) in
                guard let data = data else {
                    print("RETURNED")
                    return }

                do {
                     let decoder = JSONDecoder()
                    let rows = try decoder.decode(RootClass.self, from: data)
                    guard let JSONArt = rows.rows else { return }
                    artObjects = JSONArt
                    print("artObjects has \(artObjects.count) items")
                } catch let error {
                     print("Error: ", error)
            }
        }.resume()

    }
```

Рисунок 3.26 – клас, що обробляє JSON дані отримані з google sheets

Файл, що виконує описану команду представлено у додатку Б (див Б.8)

artObjects – змінна, в яку будуть завантажені структуровані дані у вигляді нестандартного типу даних ArtObject (рис. 3.25).

```
struct ArtObject : Codable {

    let id : Int?
    let image : String?
    let name : String?
    let authoryear : String?
    let nextconnection : String?
    let trigger1description : String?
    let trigger1name : String?
    let trigger2description : String?
    let trigger2name : String?
    let trigger3description : String?
    let trigger3name : String?
    let trigger4description : String?
    let trigger4name : String?
    let trigger5description : String?
    let trigger5name : String?
```

Рисунок 3.27 – структура нестандартного типу даних *ArtObject*

Файл, що виконує описану команду представлено у додатку Б (див Б.9)

Використання такого сервісу та збереження даних у Google Sheets дає можливість змінювати наповнення застосунку без необхідності оновлювати код застосунку.

Висновки до розділу:

У третьому розділі розроблено сценарій роботи застосунку в AR, що базується на використанні сцен. Виконано реалізацію сервісу для оновлення інформації музейним співробітниками, що не знають мови програмування.

Запропонована структура сцен та переходів між ними за допомогою відстежування картин. Вказана структура протестована та показала себе як найоптимальніша та швидка. Застосунок не робить пауз при зміні сцен.

Користувацький інтерфейс поверх AR-сцен реалізовано через сторіборди шляхом розміщення AR-сцен під сторібордом з прозорим фоном.

Використання ресурсу gsx2json дає можливість змінювати наповнення застосунку без необхідності оновлювати код.

4 СТАРТАП-ПРОЕКТ

4.1. Опис ідеї проекту

Таблиця 4.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Використання універсального рішення для створення екскурсійно-навігаційних застосунків	1. Музеї/галереї	Можливість реалізації AR застосунку для інституції без навичок програмування
	2. Навігація у місті	Проведення екскурсій в місті, можливість створення персоналізованих екскурсій

Таблиця 4.2 – Визначення характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Запропонований метод	Augmania	Visionar			
1.	Створення AR програм без навичок програмування	Дає змогу	Дає змогу	Дає змогу		+	
2.	Створення застосунку на мобільні платформи	Дає змогу	Не дає змогу	Дає змогу			+
3.	Вартість на місяць	200 €	200-1000 €	300-500 €			+

4.2. Технологічний аудит ідеї проекту.

У таблиці 4.3 показано оцінку технологічної здійсненності ідеї проекту та наведено технології, що можуть бути використані для реалізації проекту.

Таблиця 4.3. Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Універсальне рішення для створення екскурсійно-навігаційних застосунків	Власне SDK для створення інших застосунків	Необхідно перетворити створений застосунок у SDK	Доступна
2		Сервер, на якому будуть опрацьовуватись запити від інших застосунків	Необхідно розробити	Доступна

Обрана технологія реалізації ідеї проекту: універсальне рішення для створення екскурсійно-навігаційних застосунків.

4.3. Аналіз ринкових можливостей запуску стартап-проекту

У таблиці 4.4 показано попередню характеристику потенційного ринку стартап-проекту.

Таблиця 4.4. Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	6
2	Загальний обсяг продаж, грн/ум.од	500000
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Зацікавлення потенційних клієнтів
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі (або по ринку), %	500000/210000 = 238%

У таблиці 4.5 показано характеристику потенційних клієнтів стартап-проекту.

Таблиця 4.5. Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Вимоги споживачів до товару
1	Здешевлення процесу створення AR застосунків,	Інституції, що проводять екскурсії (музеї, галереї, тощо)	Легкість створення екскурсій та набуття популярності
2	Пришвидшення процесу захоплення руху	Люди які бажають ділитися цікавими екскурсіями містом	

У табл. 4.6 показані фактори загроз реалізації стартап-проекту.

Таблиця 4.6. Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Незацікавленість клієнтів	Внаслідок невдалого маркетингу клієнт може не зацікавитись послугами	Внесення застосункових сервісних послуг, демонстрація можливостей
2	Втрата конкуренції	Втрата рангу надійного поставника	Якісне та кількісне нарощування інтенсивності та грамотна цінова політика

У табл.4.7 показано фактори можливостей при реалізації стартап- проекту.

Таблиця 4.7. Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Перехід до домінування на ринку медійних послуг	Зростання попиту	Якісне та кількісне нарощування потужностей
2	Імплементація технологій в існуючі екскурсійні застосунки для розширення їх можливостей	Зростання попиту внаслідок зростання клієнтів	Якісне та кількісне нарощування потужностей

У таблиці 4.8 визначено особливості конкурентного середовища та його вплив на впровадження проекту.

Таблиця 4.8. Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1.Чиста конкуренція	Використання схожих технологій	Стандартизація на високому рівні
2.Локальний	Відсутність єдиного постачальника послуг	Окремий підхід до кожного клієнту. Технічна підтримка
3.Міжгалузева	Відсутня	Відсутня
4.Товарно-видова	Застосування стандартизованих технологій	За необхідності, використання загальновживаних апаратних та програмних засобів
5.Цінова	Наймання розробників чи компаній, які мають значну ціну	Можливість заощадити за допомогою застосування універсального рішення
6.Марочна	Для кожного застосунку потрібна команда розробників	Отримання переваги на ринку медійних послуг

У табл. 4.9 показано фактори конкурентноспроможності та їх обґрунтування.

Таблиця 4.9. Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Раціональніший ціновий показник	Відсутність необхідності виділення розробників на кожний проект.
2	Надання сервісних послуг	При бажанні клієнта, можлива технічна підтримка штатного розробника

У табл. 4.10 наведено сильні та слабкі сторони проекту.

Таблиця 4.10. Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні						
			–3	–2	–1	0	+1	+2	+3
1	Раціональніший ціновий показник	17	+						
2	Надання сервісних послуг	12		+					
3	Необхідність самостійної роботи клієнта для створення застосунку	4				+			

У табл.4.11 наведено SWOT-аналіз стартап-проекту.

Таблиця 4.11. SWOT- аналіз стартап-проекту

Сильні сторони: раціональний ціновий показник, надання сервісних послуг	Слабкі сторони: Необхідність самостійної роботи клієнта для створення застосунку
Можливості: Перехід до ексклюзивного застосування нового методу, Імплементація методу в існуючі рішення AR застосунків	Загрози: Незацікавленість клієнтів, втрата авторитету

Альтернативи ринкового впровадження стартап-проекту наведені у табл.4.12.

Таблиця 4.12. Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Укладення договорів з медійними компаніями та швидке захоплення ринку при використанні нового рішення	висока	незначні
2	Використання приладів загального вжитку для підвищення конкурентноспроможності	середня	незначні

Обрана альтернатива - укладення договорів з медійними компаніями та швидке захоплення ринку при використанні нового рішення.

4.4. Розроблення ринкової стратегії проекту

Обґрунтування вибору цільових груп потенційних споживачів наведено у табл. 4.13.

Таблиця 4.13. Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Інституції, що проводять екскурсії (музеї, галереї, тощо)	Середня	Високий	Середня	Висока
2	Люди які бажають ділитися цікавими екскурсіями містом	Висока	Високий	Середня	Низька

Визначення базової стратегії розвитку наведено у табл. 4.14.

Таблиця 4.14. Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
1	Використання альтернативних технологій та пристроїв	Встановлення нового стандарту якості	Зацікавлення та залучення найбільш популярних музеїв/галерей	Стратегія диференціації
2	Дешевизна проекту	Раціональніші витрати на обладнання, та послуги	Застосування загальновживаних апаратних рішень замість спеціалізованих комплексів	Стратегія лідерства по витратах

Визначення базової стратегії конкурентної поведінки наведено у табл.4.15.

Таблиця 4.15. Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
1	Так	Забирати існуючих та шукати нових	Не буде	Стратегія виклику лідера

Визначення стратегії позиціонування наведено у табл. 4.16.

Таблиця 4.16. Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Висока якість послуг	Стратегія диференціації	Новизна, гарант якості, точність дослідження	Якість, надійність, точність
2	Мінімальні витрати	Стратегія лідерства по витратах	Універсальність запропонованого рішення	Дешевизна, універсальність

4.5. Розроблення маркетингової програми стартап-проекту

Ключові переваги концепції потенційного товару наведено у табл. 4.17.

Таблиця 4.17. Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Якість	Висока якість, надійність	Надійність
2	Дешевизна	Раціональне використання коштів, дешевше обладнання	Дешевизна

Визначення меж встановлення ціни на послугу наведено у табл. 4.18.

Таблиця 4.18. Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	2500 у.о./од.	1800 у. о./од	Високий	Н.500 у.о. – В.1000 у.о. (Товар) Н.200 у.о. – В.500 у.о. (Послуга)

Формування системи збуту послуги наведено у табл. 4.19.

Таблиця 4.19. Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Орієнтована на отримання максимальної якості та легкості створення AR застосунків	Поставки якісного та надійного товару	Значна	Договірна система збуту

Концепції маркетингових комунікацій наведено у табл. 4.20.

Таблиця 4.20. Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Зацікавленість в якісному та легкому для розробки продукті	Медіа ресурси	Гарантованість якості та стандартизація, політика сервісності	Зацікавити у покращеннях пов’язаних із зростаючою популярністю послуг	Представлення легкості створення AR застосунків без навичок програмування
2	Зацікавленість у великій кількості продукту із дотриманням умов якісності	Медіа ресурси	Глибина каналу постачальників, гарант якості	Зацікавити у позитивних сторонах первісності та в глибині каналу постачання	

Висновки до розділу

У розділі розроблено стартап-проект, який базується на створенні універсального рішення для створення AR застосунків для інституцій, зокрема музеїв, що проводять екскурсії. Проведено дослідження доцільності та рентабельності даного бізнес-проекту та визначено, що комерціалізація проекту є доцільною.

ВИСНОВКИ

У магістерській дисертації запропоновано варіант реалізації застосунку з використанням доповненої реальності для проведення екскурсій у Національному художньому музеї України. Застосунок має можливість розвиватись до універсального рішення для проведення екскурсій у різних подібних інституціях.

В рамках магістерської дисертації було проведено дослідження методів та технологій створення доповненої реальності. На основі проведених досліджень отримано наступні результати:

1. Розглянуто існуючі засоби створення AR застосунків, проаналізовано програмні та апаратні засоби AR. Виконано порівняння найпопулярніших програмних платформ та бібліотек. Для виконання поставленого завдання обрано Torch SDK – бібліотеку, що дозволяє створювати AR без написання коду, а потім інтегрувати зроблений проект у AR додаток.
2. Розглянуто та вивчено способи переміщення об’єктів у просторі та трекінг і локалізацію об’єктів за допомогою системи SLAM і встановлено, що ця технологія стрімко розвивається, та кожна бібліотека приносить свої модифікації до існуючих рішень трекінгу.
3. Проаналізовано можливість використання бібліотеки Torch SDK для використання сцен, як окремих AR-сцен та застосовано концепцію опорних точок.
4. Проаналізовано процес експорту проекту Torch у архів формату torchkitproj. Розібрані складові цього архіву і взаємодія об’єктів та масивів всередині архіву. Розглянуто інтеграцію Torch SDK у застосунок на операційній системі iOS через менеджер залежностей CocoaPods у середовищі Xcode.
5. Виконано налаштування стандартного для операційної системи iOS ARkit під роботу з Torch SDK, та запропоновано варіант реалізації користувацького інтерфейсу поверх AR-сцен. У процесі були використані функції зворотнього зв’язку Torch SDK та налаштовано сервер на Google Drive за допомогою ресурсу gsx2json, що надало можливість музейним

співробітникам змінювати та доповнювати інформацію про експонати без необхідності оновлювати додаток та писати програмний код. Недоліком цього рішення є потреба у доступу до мережі Інтернет для отримання інформації з серверу. Але враховуючи те, що в музеї є безкоштовний WiFi – цей недолік не грає суттєвої ролі.

6. Розроблено стартап-проект, який базується на створенні універсального рішення для створення AR застосунків для інституцій, що проводять екскурсії. Проведені дослідження вказують рентабельність розробленого бізнес-проекту. Також визначено, що комерціалізація проекту є доцільною.

ПЕРЕЛІК ДЖЕРЕЛ. ПОСИЛАННЯ

1. Augmented Reality. URL: https://en.wikipedia.org/wiki/Augmented_reality
2. Sensor-Aware Recognition and Tracking for Wide-Area Augmented Reality on Mobile Phones. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4721767
3. Introducing ARkit: Augmented reality for iOS. URL: https://developer.apple.com/videos/play/wwdc2017/602/
4. Basic ARKit Concepts. URL: https://arvrjourney.com/basic-arkit-concepts-95e845fbb0a1
5. Vuforia developer library. URL: https://library.vuforia.com/getting-started/overview.html
6. Wikitude SDK documentation. URL: https://www.wikitude.com/documentation/
7. ARCore API reference. URL: https://developers.google.com/ar/reference
8. DoF, FoV, Anchors, and Markers: Demystifying Common AR Terms. URL: https://learningsolutionsmag.com/articles/dof-fov-anchors-and-markers-demystifying-common-ar-terms
9. Basics of AR: SLAM – Simultaneous Localization and Mapping. URL: https://www.andreasjakl.com/basics-of-ar-slam-simultaneous-localization-and-mapping/
10. Torch AR developer docs, getting started. URL: https://www.torch.dev/docs/ios-swift/getting-started/index.html
11. Usdz File Format Specification. URL: https://graphics.pixar.com/usd/docs/Usdz-File-Format-Specification.html
12. CocoaPods guides. URL: https://guides.cocoapods.org
13. Callback Functions. freeCodeCamp Guide. URL: https://guide.freecodecamp.org/javascript/callback-functions/
14. Google Drive. URL: https://en.wikipedia.org/wiki/Google_Drive
15. GSX2JSON - Google Spreadsheet to JSON API service. URL: http://gsx2json.com

ДОДАТОК А
ABSTRACT

Augmented reality is a field with a versatile range of applications used in many fields including recreation and education. Continually developing technology spanning the last decade has drastically improved the viability for augmented reality projects now that most of the population possesses a mobile device capable of supporting the graphic rendering systems required for them. Education in particular has benefited from these technological advances as there are now many fields of research branching into how augmented reality can be used in schools. For the purposes of Holocaust education however, there has been remarkable little research into how Augmented Reality can be used to enhance its delivery or impact. The purpose of this study is to speculate regarding the following questions: How is augmented reality currently being used to enhance history education? Does the usage of augmented reality assist in developing long-term memories? Is augmented reality capable of conveying the emotional weight of historical events? Will augmented reality be appropriate for teaching a complex field such as the Holocaust? To address these, multiple studies have been analysed for their research methodologies and how their findings may assist with the development of Holocaust education.

Museums as an augmented reality learning environment is a concept that has been explored by other researches for various means, from educational benefit to usability. A study performed by Kyriakou and Hermon analysed the usage of an augmented reality system for a museum environment. This study was performed to test user reactions when using an Augmented Reality system to examine artefacts that would otherwise be off-limits to physical interactions due to their fragility and the risk associated with allowing members of the public to touch things, such as potential damage or additional wear over time. This study utilized a mobile device with a Google Carboard head mounted display for the hardware and Unity as the engine, building a system that would allow for interaction via hand gestures. Users were tasked to perform simple operations such as grabbing an object or rotating it. The results found that an overwhelming majority of users enjoyed their experience and expressed interest in the application, with no one strongly disagreeing and only a small percentage (>10%) being neutral on the matter. There was

also mention of the learning curve required to adjust to the system but also found that users with experience playing video games were able to adjust more quickly.

The research does give brief mention to the demographics of the system users but does not analyse them further which is a topic that could be explored further; is there a correlation between age/gender group and enjoyment of using an augmented reality system?

The British Computer Society defines augmented reality as “combining the digital world with the physical one and therefore augmenting the real-world experience” [1]. This technology has rapidly become more viable for commercial and research projects in the last decade due to the prevalence of head mounted devices (HMDs) and smart devices such as phones, tablets and handheld games consoles that are now intrinsically woven into daily life. This has reduced the major challenge of deploying an augmented reality application because specialized hardware is no longer required to use the technology, instead users are able to operate the system from their own devices. With the hardware barrier reduced, augmented reality has begun seeing use for areas such as entertainment, simulations, education and training scenarios along with a variety of other applications. Regarding these educational applications however, research has been primarily focused on school education with little study on alternative audiences or environments

Usability of augmented reality devices for museums and cultural heritage sites is another area that also must be considered, as the effectiveness of the technology will be impeded if the visitor is unable to reliably control the system. Hammady and Ma researched this in their study on the creation of a virtual museum guide, specifically within the context of using the Microsoft HoloLens as their hardware. The purpose of this study was to build and test an augmented reality application on a small group of nine users to determine how they were able to use the system along with how comfortable it was. A majority of the group claimed the HoloLens was comfortable to wear, although over a third of the participants were either neutral or disagreed that they were able to look around their environment comfortably when wearing the headset. Responses to the control elements of the hardware were varied, with a majority agreeing or strongly agreeing that that they could perform the hand controls for the HoloLens whilst a few

disagreed or were neutral. The results would indicate that whilst most users were able to use the system controls, there are still concerns about comfortably wearing an HMD and being able to interact with its control system, something which the authors attributed to the bulky size of the HoloLens and its limited field of view. These concerns may possibly be alleviated with the pending release of the HoloLens 2, or potentially the use of a different HMD.

```
switch trigger.name {

case "heorhi1tapped":
    self.popUpDescription.text = artObjects[0].trigger1description
    self.presentPopUp(fromBottom: true)
case "heorhi2tapped":
    self.popUpDescription.text = artObjects[0].trigger2description
    self.presentPopUp(fromBottom: true)
case "heorhi3tapped":
    self.popUpDescription.text = artObjects[0].trigger3description
    self.presentPopUp(fromBottom: true)
case "heorhi4tapped":
    self.popUpDescription.text = artObjects[0].trigger4description
    self.presentPopUp(fromBottom: true)
case "heorhiNextTapped":
    self.popUpDescription.text = artObjects[0].nextconnection
    self.presentPopUp(fromBottom: false)
```

Figure A1 – switching triggers in an AR scene

Augmented reality usage for museums stems beyond user enjoyment, there are many elements to the use of such a system with more factors to consider. Some of these factors were identified and examined in the study performed by He et al. on their research into enhancing museum experiences. This research acknowledged how augmenting an exhibit is not limited to using 3D models, there is potential for usage of Heads-Up-Display (HUD) information similar to that in a video game. The experimental part of this study had users look at Vincent van Gogh's Starry Night painting with different augmented reality displays, some using visual elements such as making the stars glimmer or the water reflect, whilst others augmenting the scene with floating text (specified as Verbal elements), giving descriptions of the painting instead. The testers were given questionnaires in the aftermath which found that the verbal elements had a greater impact on the users and their willingness to pay for similar experiences. This shows that simply

adding visual effects to a scene is insufficient for an augmented reality system to engage with a user, the content itself must be contextually relevant and informative.

This point was preceded in another museum-based study performed by Keil et al. on designing personalized stories with handheld augmented reality devices. The intention of this study was to utilize handheld devices such as smartphones and tablets to add augmented reality to exhibits in the Acropolis Museum in Athens, Greece. The use of the augmented reality was to provide interactive storytelling experiences that were related to the exhibits and included features such as videos, games, audio narration, imagery and digital reconstructions. This study was documented as part of a conference paper and did not publish any results; however, the discussion gave mention to an evaluation of one of the stories and discussed how the visitors wanted to use the augmented reality to obtain more information about the exhibits. Without access to the data to confirm these findings, this may not be a reliable result to obtain from this study, but this point does correlate with the data obtained in the study by He et al.

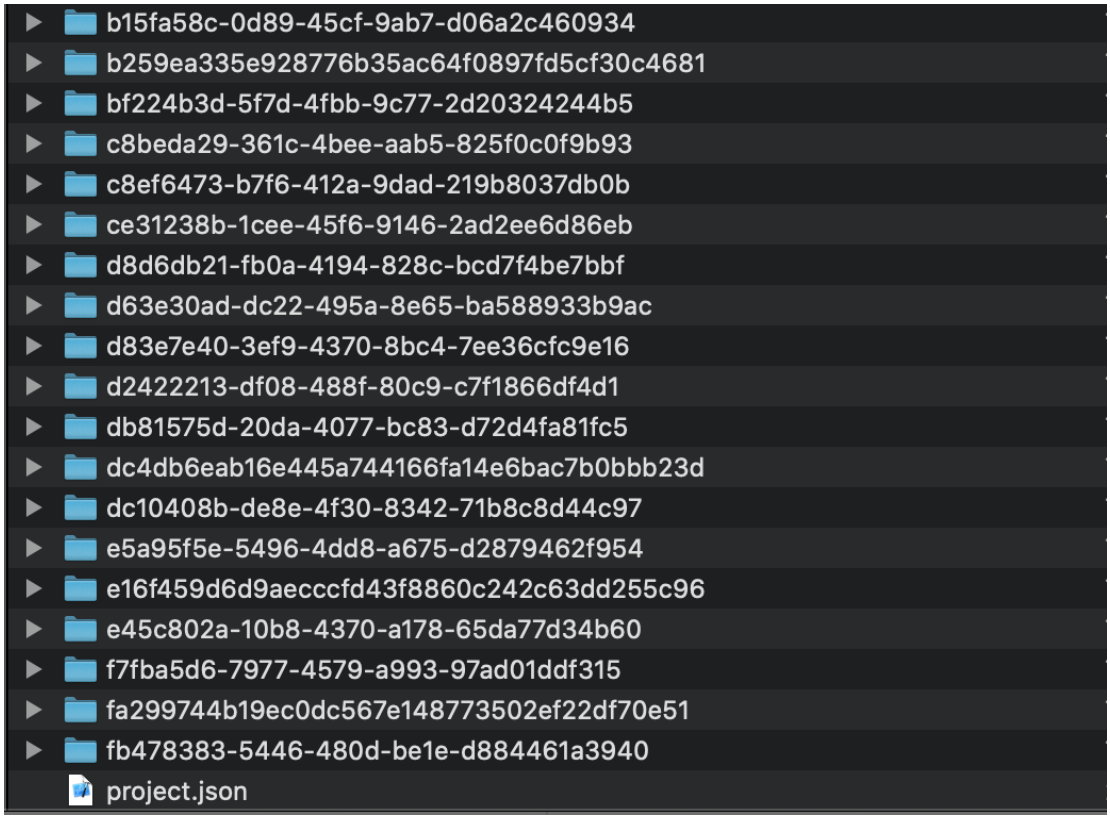


Figure A1 - correlated data obtained in the study

Science Museums have been a target of research for augmented reality research as science is an area where abstract concepts may be difficult to explain or the topic of the

exhibit is not directly observable within the museum environment. This was further explored in the research of Yoon and Wang of visualizing invisible concepts in science museums with augmented reality technology. The purpose of this study was to use augmented reality to visualize magnetic fields for teaching science to children between the ages of nine and twelve. The results found that the students in the augmented reality group were displaying more cognitive behaviours than those in the control group, including additional team working and problem identifying skills. The study also recorded the amount of time that each group spent with the exhibit and found that the students with the augmented reality system were interacting with the exhibit for longer by an average of seventy-three seconds, which would indicate that the system was not only beneficial for cognitive development and learning, but also for keeping the audience's interest.

The future of augmented, mixed and virtual reality research was discussed in a paper by Bekele et al. in their survey of these technologies for usage in cultural heritage. The survey is a very comprehensive technical analysis of the three technologies along with an examination of the hardware and software that enables them and their uses. The authors also discussed areas of the technology that research can expand upon in future studies. These were the examples provided;

1. Robust Tracking: Use camera-based tracking instead of sensor-based tracking as it's less prone to errors and has had more development.
2. Standardization: Define or adopt a standard for the creation of the system, including the mark-up language, documentation, data structures, metadata and model format. The authors claim a community standard for project creation would assist in development of future projects.
3. User-Driven Semantics: Allow the user to focus on points of interest to avoid cluttering them with information in environments with multiple areas of interest.
4. Tangible Augmented Reality: utilize tangible user interfaces, a type of user interface interfacing that allows the user to interact with and directly

manipulate the information provided by a system by interacting with physical objects in the real world.

5. Fully Immersive Virtual Reality: Whilst not applicable to this paper, the concept involves creating a fully immersive simulation for the user to enhance their experience. The authors did comment that this type of content is very expensive to manufacture.

As augmented reality can be used as a tool for gamified learning, it has the potential to assist as a motivational asset providing it meets those three criteria. Utilizing augmented reality for an educational task provides the user with total autonomy as they can progress at their own speed and investigate the elements that interest them, therefore having the potential to motivate them beyond what a traditional learning would be capable of.

Theory was put into practice during a study performed by Di Serio et al. on how augmented reality could be used to motivate middle-school students. Students who were due to begin studying renaissance era Italian art were shown traditional paintings that were enhanced with augmented reality to include text, audio, video and 3d model files. The students were then given autonomy to free explore the augmented reality elements of their class using the hardware provided, however the results collected from this study are not fully dependable as it was unable to divide the students into control or focus groups. Sixty-nine students participated but fourteen of them were deleted from the sample due to missing data or outliers. The initial testing focused on four categories; Attention, relevance, confidence and satisfaction, each measured on a scale of one to five. In the first round of tests, these categories scored below 3.5, but almost all rose above that during the second round to 3.76, 3.48, 3.63 and 3.51 respectively. The relevance score only increased by a mean value of 0.17 between tests, which would indicate that the usage of augmented reality does not increase a student’s interest in a topic.

Unfortunately, due to the lack of a control group, it is not possible to derive from this study whether the usage of augmented reality is beneficial or detrimental to motivation when compared to traditional teaching methods.

The results from previous research are highly encouraging for future studies and projects as they indicate that usage of augmented reality has been able to stimulate learning engagement in bold new ways that have previously been unavailable. Many of these studies have noted a lack of empirical research performed within the area, signifying that there is still a great deal of potential study still to be done. The results found thus far have indicated very strongly that as a technology, augmented reality has the potential to revolutionize the way that certain subjects are taught, along with being able to bring new experiences or revitalize old ones at museums, heritage sites and other areas of historical value. Historically significant events that have occurred within living memory have the rare and unique opportunity to be recorded within the confines of modern technology, such as augmented or virtual reality so that the first-hand accounts can be told or experienced directly by current and future generations. Documentaries and interview recordings existed prior to this technology emerging, but they lacked the capacity to actively engage with the delivered content.

Future research projects into the development of augmented reality applications for educational purposes should be aware of the requirements of the users to engage with a system, along with the potential hazards that they will face when interacting with it. The studies show that technology is effective for motivating learners, stimulating interest and building short term memory. We seek to utilize the knowledge gained from this review in our upcoming collaboration with the Neuengamme Holocaust Memorial Site, with whom we are collaborating to develop an augmented reality application for educational purposes.

The research focus of this study will be to expand upon study on the effect of augmented reality on developing long term memories. Whilst the theory of this has been covered in this review, there is a lack of empirical study on how effective augmented reality is for creating long term memories. The testing of this project will educate users on the Neuengamme Holocaust Memorial site using two groups; a focus group and a

control group. The focus group will have their experience enhanced with an augmented reality system and the control group will be given a more traditional class on the subject. At intervals following these classes, the groups will be given online tests to determine how much information they have retained about this part of the Holocaust and have the results compared to see which group was able to remember the most.

ДОДАТОК Б

Лістинг програми

Б.1. AppDelegate.swift

```
import UIKit
import TorchKit
import os

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?


    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[UIApplication.LaunchOptionsKey: Any]?) -> Bool {

        if #available(iOS 13.0, *) {
            window?.overrideUserInterfaceStyle = .light
        } else {
            // Fallback on earlier versions
        }

        do {
            try TorchKit.shared.initSDK(apiKey:
"djIucHVibGljLmV5SndjbTkwYnlJNklrTkJSVk5JUjNCTlVURnJlRmRZYkVaT2JXUlRVVlJrUlZGV2FFMVNSM1J
RVlZSSmVtUnJWa3ROVkVsaFJFRnFjaTlrY25KQ1VrTkJNRzlITlVGVFNWTkRhRUpDUm1wTWQyMXVlVVZ5U
1Vwa1F6WkdZbEpQVjIwaWZjVHVrTXV5cmZHY2VBS21QQVFEajdmdVlYUVhQaFgtTUlpbzR4OWRwV1gtYTl
KdHVjaVRJRUZFLW5ORlhnb2JhYzdPVm1vV2Zra0Eyc2d3MlVvTVBBQQ==")
        } catch {
            os_log(OSLogType.default, "Init failed!")
        }



        return true
    }

    func applicationWillResignActive(_ application: UIApplication) {
        // Sent when the application is about to move from active to inactive state. This can occur for certain types
of temporary interruptions (such as an incoming phone call or SMS message) or when the user quits the
application and it begins the transition to the background state.
        // Use this method to pause ongoing tasks, disable timers, and invalidate graphics rendering callbacks.
Games should use this method to pause the game.
    }

    func applicationDidEnterBackground(_ application: UIApplication) {
        // Use this method to release shared resources, save user data, invalidate timers, and store enough
application state information to restore your application to its current state in case it is terminated later.
        // If your application supports background execution, this method is called instead of
applicationWillTerminate: when the user quits.
    }

    func applicationWillEnterForeground(_ application: UIApplication) {
        // Called as part of the transition from the background to the active state; here you can undo many of the
changes made on entering the background.
    }

    func applicationDidBecomeActive(_ application: UIApplication) {
        // Restart any tasks that were paused (or not yet started) while the application was inactive. If the
application was previously in the background, optionally refresh the user interface.
    }

    func applicationWillTerminate(_ application: UIApplication) {
```

```
        TorchKit.shared.shutdownSDK()

    }


}
```

Б.2. AboutViewController

```
import UIKit

class AboutViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        // Do any additional setup after loading the view.
    }
    @IBAction func closeButtonTapped(_ sender: UIButton) {
        self.dismiss(animated: true, completion: nil)
    }
}
```

Б.3. ArtDetailViewController.swift

```
import UIKit

class ArtDetailViewController: UIViewController {

    @IBOutlet weak var artName: UILabel!
    @IBOutlet weak var artImageView: UIImageView!
    @IBOutlet weak var artDescription: UITextView!
    @IBOutlet weak var artistYearLabel: UILabel!
    @IBOutlet weak var artHeightConstraint: NSLayoutConstraint!
    var descriptionText = ""
    var imageURL = ""
    var artNameText = ""
    var author_yearText = ""
    override func viewDidLoad() {
        super.viewDidLoad()

        setImage()
        artDescription.text = descriptionText
        artName.text = "«\(artNameText)»"
        artistYearLabel.text = author_yearText

    }


    func setImage() {
        artImageView.sd_setImage(with: URL(string: imageURL), placeholderImage: UIImage(named:
"loadingIcon"), completed: { (image, error, cacheType, imageURL) in
            let downImage = image as! UIImage
            //self.artImageHeight.constant = downImage.size.height
            let ratio = downImage.size.width / downImage.size.height
```

```
            let newHeight = self.artImageView.frame.width / ratio
            self.artHeightConstraint.constant = newHeight
            //self.artImageView.layoutIfNeeded()
            self.view.layoutIfNeeded()
        })

    }

    @IBAction func closeButtonTapped(_ sender: UIButton) {
        self.dismiss(animated: true, completion: nil)
    }

}
```

Б.4. ArtsViewController.swift

```
import UIKit
import SDWebImage

class ArtsViewController: UIViewController {

    @IBOutlet weak var collectionView: UICollectionView!

    let popUpContentManager = PopUpContentManager.shared

    override func viewDidLoad() {
        super.viewDidLoad()

        collectionView.delegate = self
        collectionView.dataSource = self

        collectionView.register(UINib(nibName: "CollectionViewCell", bundle: nil), forCellWithReuseIdentifier:
"cell")
    }

    @IBAction func backButtonPressed(_ sender: UIButton) {
        self.dismiss(animated: true, completion: nil)
    }

}


extension ArtsViewController: UICollectionViewDelegate, UICollectionViewDataSource,
UICollectionViewDelegateFlowLayout {

    func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection section: Int) -> Int {
        return artObjects.count
    }

    func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout:
UICollectionViewLayout, sizeForItemAt indexPath: IndexPath) -> CGSize {
        let flowayout = collectionViewLayout as? UICollectionViewFlowLayout
        let space: CGFloat = (flowayout?.minimumInteritemSpacing ?? 0.0) + (flowayout?.sectionInset.left ?? 0.0)
+ (flowayout?.sectionInset.right ?? 0.0)
        let size:CGFloat = (collectionView.frame.size.width - space) / 2.0
        return CGSize(width: size, height: size)
    }
```

```
    func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath) ->
UICollectionViewCell {
        let row = artObjects[indexPath.row]
        let cell = collectionView.dequeueReusableCell(withReuseIdentifier: "cell", for: indexPath) as!
CustomCollectionViewCell
        cell.isUserInteractionEnabled = false

        if let artURL = row.image {
            cell.artImageView.sd_setImage(with: URL(string: artURL), placeholderImage: UIImage(named:
"loadingIcon"), completed: { (image, error, cacheType, imageURL) in
                cell.isUserInteractionEnabled = true
            })
        }

        return cell
    }

    func collectionView(_ collectionView: UICollectionView, didSelectItemAt indexPath: IndexPath) {
        let vc = storyboard?.instantiateViewController(withIdentifier: "ArtDetailViewController") as?
ArtDetailViewController


        let rawTexts = [artObjects[indexPath.row].trigger1description,
artObjects[indexPath.row].trigger2description, artObjects[indexPath.row].trigger3description,
artObjects[indexPath.row].trigger4description, artObjects[indexPath.row].trigger5description]

        var description = ""
        for item in rawTexts {
            if item != "nil" {
                description = description + " " + item!
            }
        }

        vc?.artNameText = artObjects[indexPath.row].name!
        vc?.descriptionText = description
        vc?.imageURL = artObjects[indexPath.row].image!
        vc?.author_yearText = artObjects[indexPath.row].authoryear!

        vc!.modalPresentationStyle = .fullScreen
        self.present(vc!, animated: true, completion: nil)

    }

}
```

Б.5. ContactsViewController.swift

```
import UIKit
import MessageUI

class ContactsViewController: UIViewController, MFMailComposeViewControllerDelegate {

    override func viewDidLoad() {
        super.viewDidLoad()

        // Do any additional setup after loading the view.
    }
```

```
    func makePhoneCall(phoneNumber: String) {
        if let phoneURL = NSURL(string: ("tel://" + phoneNumber)) {

                UIApplication.shared.open(phoneURL as URL, options: [:], completionHandler: nil)

        }
    }
    @IBAction func adressPressed(_ sender: UIButton) {
        if (UIApplication.shared.canOpenURL(URL(string:"comgooglemaps://")!)) {
          UIApplication.shared.openURL(URL(string:

"comgooglemaps://?daddr=National+Art+Museum+of+Ukraine,+6,+Mykhaila+Hrushevskoho+St,+Kyiv,+01001")
!)
        } else {
            UIApplication.shared.openURL(URL(string:"https://goo.gl/maps/enwEkMRRYxwUiu7Q7")!)
          print("Can't use comgooglemaps://");
        }
    }

    @IBAction func phone1Pressed(_ sender: UIButton) {
        makePhoneCall(phoneNumber: "+380442781357")
    }

    @IBAction func phone2Pressed(_ sender: UIButton) {
        makePhoneCall(phoneNumber: "+380442787454")
    }


    @IBAction func emailPressed(_ sender: UIButton) {
        if MFMailComposeViewController.canSendMail() {

            let alert = UIAlertController(title: "Надіслати лист у NAMU", message: nil, preferredStyle: .alert)
            alert.addAction(UIAlertAction(title: "Так", style: .default, handler: { (action) in
                let mail = MFMailComposeViewController()
                mail.mailComposeDelegate = self
                mail.setToRecipients(["info@namu.kiev.ua"])
                //mail.setSubject("Poof feedback")

                self.present(mail, animated: true)
            }))

            alert.addAction(UIAlertAction(title: "Cancel", style: .cancel, handler: nil))

            self.present(alert, animated: true, completion: nil)

        } else {
            return
        }
    }


    @IBAction func fbButtonPressed(_ sender: UIButton) {
        guard let url = URL(string: "https://www.facebook.com/namu.museum/") else { return }
        UIApplication.shared.open(url)
    }


    @IBAction func instaButtonPressed(_ sender: UIButton) {
        guard let url = URL(string: "https://www.instagram.com/namu.museum/") else { return }
        UIApplication.shared.open(url)
    }
```

```
    @IBAction func websiteButtonPressed(_ sender: UIButton) {
        guard let url = URL(string: "http://namu.kiev.ua/ua/about/history.html") else { return }
        UIApplication.shared.open(url)
    }


    @IBAction func backButtonPressed(_ sender: UIButton) {
        self.dismiss(animated: true, completion: nil)
    }


}
```

Б.6. OtherViewController.swift

```
import UIKit

protocol TorchProjectDelegate {
    func resetTorchProj()
}

class OtherViewController: UIViewController {

    var torchProjDelegate: TorchProjectDelegate?

    @IBOutlet weak var tableView: UITableView!

    let onboardingVC = OnboardingViewController.shared

    let sectionItems = ["Розпочати AR екскурсію з початку", "Як користуватися застосунком", "Список
картин", "Модель церкви Вознесіння", "Мапа музею", "Контакти", "Залишити фідбек", "Про застосунок"]

    override func viewDidLoad() {
        super.viewDidLoad()


        tableView.delegate = self
        tableView.dataSource = self
        tableView.separatorStyle = .singleLine
        tableView.separatorColor = .black
        tableView.tableFooterView = UIView()

        let px = 1 / UIScreen.main.scale
        let frame = CGRect(x: 0, y: 0, width: self.tableView.frame.size.width, height: px)
        let line = UIView(frame: frame)
        self.tableView.tableHeaderView = line
        line.backgroundColor = self.tableView.separatorColor

        tableView.rowHeight = 80
    }




    @IBAction func backButtonPressed(_ sender: UIButton) {
        self.dismiss(animated: true, completion: nil)
    }
```

```
    func presentOnboardingRestartAlert() {
        let alert = UIAlertController(title: "Як користуватися застосунком", message: "Пройти ґайд ще раз?",
preferredStyle: UIAlertController.Style.alert)

        alert.addAction(UIAlertAction(title: "Так", style: UIAlertAction.Style.default, handler: { action in
            let onbVC = self.onboardingVC.createOnboardingVC()
            onbVC.modalPresentationStyle = .overFullScreen
            onbVC.presentFrom(self, animated: true)
        }))
        alert.addAction(UIAlertAction(title: "Скасувати", style: UIAlertAction.Style.cancel, handler: nil))

        self.present(alert, animated: true, completion: nil)
    }

    func presentTorchProjRestartAlert() {
        let alert = UIAlertController(title: "Почати екскурсію з початку?", message: nil, preferredStyle:
UIAlertController.Style.alert)

        alert.addAction(UIAlertAction(title: "Так", style: UIAlertAction.Style.default, handler: { action in
            self.torchProjDelegate?.resetTorchProj()
        }))
        alert.addAction(UIAlertAction(title: "Скасувати", style: UIAlertAction.Style.cancel, handler: nil))

        self.present(alert, animated: true, completion: nil)
    }


}

extension OtherViewController: UITableViewDelegate, UITableViewDataSource {

    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return sectionItems.count
    }


    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {

        let cell = tableView.dequeueReusableCell(withIdentifier: "cell", for: indexPath)
        cell.textLabel?.font = UIFont(name: "NAMU-1960", size: 16.0)
        cell.textLabel?.text = sectionItems[indexPath.row]

        return cell
    }


    func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {

        switch indexPath.row {
        case 0:
            print("start over")
            presentTorchProjRestartAlert()
        case 1:
            print("onboarding restart")
            presentOnboardingRestartAlert()
        case 2:
            print("arts")
            performSegue(withIdentifier: "toArts", sender: self)
        case 3:
            print("church 3D")
            performSegue(withIdentifier: "toChurch", sender: self)
        case 4:
```

```
            print("3d map")
            performSegue(withIdentifier: "toMap", sender: self)
        case 5:
            print("contacts")
            performSegue(withIdentifier: "toContacts", sender: self)
        case 6:
            print("feedback")
        case 7:
            print("About")
            performSegue(withIdentifier: "toAbout", sender: self)
        default:
            break
        }

        tableView.deselectRow(at: indexPath, animated: true)
    }

}
```

Б.7. TorchProjectViewController.swift

```
import ARKit
import Metal
import SceneKit
import TorchKit
import UIKit
import OnboardKit
import SwiftEntryKit
import Network


class TorchProjectViewController: UIViewController, ARSCNViewDelegate, ARSessionDelegate {


  @IBOutlet var sceneView: ARSCNView!
  @IBOutlet var sessionInfoLabel: UILabel?
    @IBOutlet weak var menuButton: UIButton!
    @IBOutlet weak var popUpView: UIView!
    @IBOutlet weak var popUpDescription: UILabel!

    @IBOutlet weak var notificationView: UIView!
    @IBOutlet weak var notificationText: UILabel!

    @IBOutlet weak var guideView: UIView!
    @IBOutlet weak var guideTopLabel: UILabel!
    @IBOutlet weak var guideBottomLabel: UILabel!
    @IBOutlet weak var guideImageView: UIImageView!

    @IBOutlet weak var visualEffectView: UIVisualEffectView!

    //Monitor for checking internet connection
    let monitor = NWPathMonitor()

    let onboardingVC = OnboardingViewController.shared
    let popUpContentManager = PopUpContentManager.shared

 //MARK: Project URL
  var projectURL = Bundle.main.url(forResource: "NAMU-14", withExtension: "torchkitproj")!
```

```
    var projectIsLoaded = false
    var loadedProject: TorchProjectNode?


  private var torchProject: TorchProjectNode?
  private var lastTime: TimeInterval = 0.0
  private var projectAnchorManager: ProjectAnchorManager?
  private var viewLock: NSLock = NSLock()
  private var viewCenter: CGPoint = CGPoint(x: 0, y: 0)
  private var lastSize: CGSize = CGSize(width: 0, height: 0)




    //MARK: ViewDidLoad
  override func viewDidLoad() {
    super.viewDidLoad()
    self.sceneView = ARSCNView(frame: self.view.bounds)

    //Run the monitor to catch connection changes
    let queue = DispatchQueue(label: "Monitor")
    monitor.start(queue: queue)

    popUpContentManager.loadData()
    print(artObjects.count)

    //ADD SCENEVIEW BELOW BUTTON

    self.view.insertSubview(self.sceneView, belowSubview: menuButton)

    self.sessionInfoLabel = UILabel(frame: CGRect(x: 0, y: 75, width: self.view.bounds.width * 0.8, height: 100))
    self.sessionInfoLabel!.numberOfLines = 3
    self.sessionInfoLabel!.textColor = .white
    self.sessionInfoLabel!.shadowColor = .black
    self.sessionInfoLabel!.shadowOffset = CGSize(width: 1.0, height: 1.0)
    self.view.addSubview(self.sessionInfoLabel!)
    self.sceneView.translatesAutoresizingMaskIntoConstraints = false
    self.sceneView.widthAnchor.constraint(equalTo: self.view.widthAnchor).isActive = true
    self.sceneView.heightAnchor.constraint(equalTo: self.view.heightAnchor).isActive = true
    self.sessionInfoLabel!.translatesAutoresizingMaskIntoConstraints = false
    self.sessionInfoLabel!.centerXAnchor.constraint(equalTo: self.view.centerXAnchor).isActive = true
    self.sessionInfoLabel!.widthAnchor.constraint(equalTo: self.view.widthAnchor, multiplier: 0.8).isActive = true
    self.sessionInfoLabel!.topAnchor.constraint(equalTo: self.sceneView.topAnchor, constant: 75.0).isActive =
true
    self.sessionInfoLabel!.heightAnchor.constraint(equalToConstant: 180.0)
    self.sessionInfoLabel?.font = UIFont(name: "NAMU-1960", size: 16.0)

  }

    //MARK: ViewDidAppear
  override func viewDidAppear(_ animated: Bool) {
    super.viewDidAppear(animated)


    if projectIsLoaded == false {
        let configuration = ARWorldTrackingConfiguration()
        //PLANE DETECTION
        configuration.planeDetection = [.horizontal]
        self.sceneView.session.run(configuration)
        self.sceneView.session.delegate = self
        UIApplication.shared.isIdleTimerDisabled = true

        //MARK: TorchProject loading
```

```
        do {
          self.torchProject = try TorchProjectNode(withProjectURL: self.projectURL, andDevice:
self.sceneView.device!, arSession: self.sceneView.session)
        } catch {
          fatalError(error.localizedDescription)
        }

        guard let torchProj = self.torchProject else { return }
        loadedProject = torchProj
        TorchGestureManager.shared.addGestureRecognizers(to: self.sceneView)
        self.sceneView.session.delegate = torchProj.arSessionDelegate ?? nil
        self.setupSceneLighting()
        self.sceneView.scene.rootNode.addChildNode(torchProj)
        self.projectAnchorManager = nil
        self.sceneView.delegate = self

        projectIsLoaded = true


         //DEBUG SETTING SCENE
        loadedProject?.setScene(sceneName: "to drunk", resetCurrentScene: true)
    }


    if UserDefaults.standard.bool(forKey: "OnboardingComplete") == false {
        let onbVC = onboardingVC.createOnboardingVC()
        onbVC.modalPresentationStyle = .overFullScreen
        onbVC.presentFrom(self, animated: true)
    }

    //Catch network changes
    monitor.pathUpdateHandler = { path in
        if path.status == .satisfied {
            print("We're connected!")
            self.UI() {
                self.removeNetworkNotification()
            }
        } else {
            print("No connection.")
            self.UI() {
                self.presentNetworkNotification()
            }
        }
    }


    //MARK: Catching triggers

    print("CURRENT SCENE IS \(loadedProject!.currentScene.name)")

    torchProject?.triggerFired = { trigger in

        switch trigger.name {

        case "heorhi1tapped":
            self.popUpDescription.text = artObjects[0].trigger1description
            self.presentPopUp(fromBottom: true)
        case "heorhi2tapped":
            self.popUpDescription.text = artObjects[0].trigger2description
            self.presentPopUp(fromBottom: true)
        case "heorhi3tapped":
            self.popUpDescription.text = artObjects[0].trigger3description
            self.presentPopUp(fromBottom: true)
```

```
case "heorhi4tapped":
    self.popUpDescription.text = artObjects[0].trigger4description
    self.presentPopUp(fromBottom: true)
case "heorhiNextTapped":
    self.popUpDescription.text = artObjects[0].nextconnection
    self.presentPopUp(fromBottom: false)

case "horse1tapped":
    self.popUpDescription.text = artObjects[1].trigger1description
    self.presentPopUp(fromBottom: true)
case "horse2tapped":
    self.popUpDescription.text = artObjects[1].trigger2description
    self.presentPopUp(fromBottom: true)
case "horse3tapped":
    self.popUpDescription.text = artObjects[1].trigger3description
    self.presentPopUp(fromBottom: true)
case "horse4tapped":
    self.popUpDescription.text = artObjects[1].trigger4description
    self.presentPopUp(fromBottom: true)
case "horse5tapped":
    self.popUpDescription.text = artObjects[1].trigger5description
    self.presentPopUp(fromBottom: true)
case "horseNextTapped":
    self.popUpDescription.text = artObjects[1].nextconnection
    self.presentPopUp(fromBottom: false)

case "velyk1tapped":
    self.popUpDescription.text = artObjects[2].trigger1description
    self.presentPopUp(fromBottom: true)
case "velyk2tapped":
    self.popUpDescription.text = artObjects[2].trigger2description
    self.presentPopUp(fromBottom: true)
case "velyk3tapped":
    self.popUpDescription.text = artObjects[2].trigger3description
    self.presentPopUp(fromBottom: true)
case "velykNextTapped":
    self.popUpDescription.text = artObjects[2].nextconnection
    self.presentPopUp(fromBottom: false)

case "ded1tapped":
    self.popUpDescription.text = artObjects[3].trigger1description
    self.presentPopUp(fromBottom: true)
case "ded2tapped":
    self.popUpDescription.text = artObjects[3].trigger2description
    self.presentPopUp(fromBottom: true)
case "ded3tapped":
    self.popUpDescription.text = artObjects[3].trigger3description
    self.presentPopUp(fromBottom: true)
case "dedNextTapped":
    self.popUpDescription.text = artObjects[3].nextconnection
    self.presentPopUp(fromBottom: false)

case "mamay1tapped":
    self.popUpDescription.text = artObjects[4].trigger1description
    self.presentPopUp(fromBottom: true)
case "mamay2tapped":
    self.popUpDescription.text = artObjects[4].trigger2description
    self.presentPopUp(fromBottom: true)
case "mamay3tapped":
    self.popUpDescription.text = artObjects[4].trigger3description
    self.presentPopUp(fromBottom: true)
case "mamay4tapped":
    self.popUpDescription.text = artObjects[4].trigger4description
```

```
    self.presentPopUp(fromBottom: true)
case "mamayNextTapped":
    self.popUpDescription.text = artObjects[4].nextconnection
    self.presentPopUp(fromBottom: false)

case "babushka1tapped":
    self.popUpDescription.text = artObjects[5].trigger1description
    self.presentPopUp(fromBottom: true)
case "babushka2tapped":
    self.popUpDescription.text = artObjects[5].trigger2description
    self.presentPopUp(fromBottom: true)
case "babushkaNextTapped":
    self.popUpDescription.text = artObjects[5].nextconnection
    self.presentPopUp(fromBottom: false)

case "drunk1tapped":
    self.popUpDescription.text = artObjects[6].trigger1description
    self.presentPopUp(fromBottom: true)
case "drunk2tapped":
    self.popUpDescription.text = artObjects[6].trigger2description
    self.presentPopUp(fromBottom: true)
case "drunkNextTapped":
    self.popUpDescription.text = artObjects[6].nextconnection
    self.presentPopUp(fromBottom: false)

case "monk1tapped":
    self.popUpDescription.text = artObjects[7].trigger1description
    self.presentPopUp(fromBottom: true)
case "monk2tapped":
    self.popUpDescription.text = artObjects[7].trigger2description
    self.presentPopUp(fromBottom: true)
case "monkNextTapped":
    self.popUpDescription.text = artObjects[7].nextconnection
    self.presentPopUp(fromBottom: false)

case "love1tapped":
    self.popUpDescription.text = artObjects[8].trigger1description
    self.presentPopUp(fromBottom: true)
case "love2tapped":
    self.popUpDescription.text = artObjects[8].trigger2description
    self.presentPopUp(fromBottom: true)
case "loveNextTapped":
    self.popUpDescription.text = artObjects[8].nextconnection
    self.presentPopUp(fromBottom: false)

case "forest1tapped":
    self.popUpDescription.text = artObjects[9].trigger1description
    self.presentPopUp(fromBottom: true)
case "forest2tapped":
    self.popUpDescription.text = artObjects[9].trigger2description
    self.presentPopUp(fromBottom: true)
case "forestNextTapped":
    self.popUpDescription.text = artObjects[9].nextconnection
    self.presentPopUp(fromBottom: false)

case "winter1tapped":
    self.popUpDescription.text = artObjects[10].trigger1description
    self.presentPopUp(fromBottom: true)
case "winter2tapped":
    self.popUpDescription.text = artObjects[10].trigger2description
    self.presentPopUp(fromBottom: true)
case "winterNextTapped":
    self.popUpDescription.text = artObjects[10].nextconnection
```

```
            self.presentPopUp(fromBottom: false)

        case "triggerChurchNotif":
            self.notificationText.text = "Подивись на ікону попереду!"
            self.presentNotification()

        case "church1tapped":
            self.popUpDescription.text = "Церква Воздвиження - яскравий приклад української дерев'яної архітектури доби зрілого бароко. Була споруджена на місці старої церкви у 1759(61) році на замовлення громади чернігівської губернії. До нашого часу дійшли небагато церков такого типу через антирелігійну політику радянського союзу. Церква, що перед вами, була розібрана у 30-х роках 20 ст. Перед самим її знесенням дослідники зробили детальні заміри, замальовки та фото, щоб зберегти інформацію для майбутніх поколінь. Саме завдяки цим даним була створена 3д-модель."
            self.presentPopUp(fromBottom: true)
        case "church2tapped":
            self.popUpDescription.text = "Іконостас - це, буквально, стіна ікон, що відокремлює сакральну(вівтарну) частину церкви від прихожан. З 16-го ст. виникає класичний високий іконостас, який складався з 5-ти основних рядів або чинів (знизу вгору): Цокольний, де переважно зображували біблейські сцени Старого Заповіту. Намісний - парні ікони Ісуса Христа та Богоматері, а також святих, які пов'язані із  церквою. Празниковий, присвячений церковним святам Деісусний - головний чин іконостасу, на якому зображено Спасителя, Богоматір, Івана Предтечу та ряд апостолів. Та пророчий ряд. Ікони були збережені завдяки Жольлвському, який вивіз їх з храму до його зруйнування."
            self.presentPopUp(fromBottom: true)
        case "churchNextTapped":
            self.popUpDescription.text = "Окрім релігійного мистецтва у період бароко активно починає розвиватись і мистецтво світське."
            self.presentPopUp(fromBottom: false)


        case "fondsTapped":
            self.guideImageView.image = UIImage(named: "thanksGuy")
            self.guideTopLabel.text = "Сподіваємось, було цікаво :)"
            self.guideBottomLabel.text = "Ми старались, аби застосунок був цікавим для вас. Але ви можете зробити його ще кращим ;) Для цього потрібно залишити свій відгук!"
            self.guideBottomLabel.font = UIFont(name: "NAMU-1960", size: 11.0)
            self.view.insertSubview(self.guideView, aboveSubview: self.sceneView)
            self.guideView.center = self.view.center
            self.guideView.transform = CGAffineTransform.init(scaleX: 1.3, y: 1.3)
            self.guideView.alpha = 0
                UIView.animate(withDuration: 0.5) {
                    self.guideView.alpha = 1
                    self.guideView.transform = CGAffineTransform.identity
                }

            let tap = UITapGestureRecognizer(target: self, action: #selector(self.handleThanksTap(_:)))
            self.guideView.addGestureRecognizer(tap)
            self.guideView.isUserInteractionEnabled = true


        default:
            break
        }


        print("Trigger \(trigger.name) fired")
        print("CURRENT SCENE IS \(self.loadedProject!.currentScene.name)")
    }

}

    //MARK: ViewDidDisappear
override func viewWillDisappear(_ animated: Bool) {
    super.viewWillDisappear(animated)
```

```
  // DID SOME SHIT
  //self.sceneView.session.pause()
}
  //MARK: ViewDidLayoutSubviews
override func viewDidLayoutSubviews() {
  super.viewDidLayoutSubviews()
  // This gets called alot, use lastSize to only lock when needed.
  if self.view.frame.size != self.lastSize {
    self.viewLock.lock()
    self.viewCenter = CGPoint(x: self.view.frame.size.width * 0.5, y: self.view.frame.size.height * 0.5)
    self.viewLock.unlock()
    self.lastSize = self.view.frame.size
  }
}

  //MARK: Renderer
func renderer(_ renderer: SCNSceneRenderer, updateAtTime time: TimeInterval) {
  guard self.projectAnchorManager == nil,
    let camTransform = self.sceneView.session.currentFrame?.camera.transform else {
    return
  }
  let dt = self.lastTime != 0 ? time - self.lastTime : 0
  self.lastTime = time
  self.viewLock.lock()
  let viewCenter = self.viewCenter
  self.viewLock.unlock()
  let hits = self.sceneView.hitTest(viewCenter, options: nil)
  let gazedNode: SCNNode? = !hits.isEmpty ? hits.first!.node : nil
  self.torchProject!.tick(delta: dt, cameraTransform: camTransform, currentGazedNode: gazedNode)
}
  //MARK: SceneLightning
func setupSceneLighting() {
  let envmapURL = Bundle.main.url(forResource: "envmap", withExtension: "png")!
  let envmapData = try! Data(contentsOf: envmapURL)
  let envmap = UIImage(data: envmapData)

  sceneView.automaticallyUpdatesLighting = false
  self.sceneView.autoenablesDefaultLighting = false
  self.sceneView.scene.lightingEnvironment.contents = envmap
  self.sceneView.scene.lightingEnvironment.intensity = 1.0
  let worldPlaneGeometry = SCNFloor()
  let worldPlaneNode = SCNNode(geometry: worldPlaneGeometry)
  let worldPlaneMaterial = SCNMaterial()
  worldPlaneGeometry.reflectivity = 0
  worldPlaneMaterial.diffuse.contents = UIColor.white
  worldPlaneMaterial.lightingModel = .physicallyBased
  worldPlaneMaterial.writesToDepthBuffer = true
  worldPlaneMaterial.colorBufferWriteMask = []
  worldPlaneGeometry.materials = [worldPlaneMaterial]
  worldPlaneNode.castsShadow = false
  //comment next line
  //self.sceneView.scene.rootNode.addChildNode(worldPlaneNode)
  let light = SCNLight()
  light.type = .directional
  light.shadowMode = .deferred
  light.intensity = 1000.0
  light.color = UIColor(white: 1.0, alpha: 1.0)
  //comment next line
  light.castsShadow = true
  light.shadowColor = UIColor.black.withAlphaComponent(0.5)
  light.shadowBias = 32
  light.shadowSampleCount = 4
  light.shadowRadius = 5.0
```

```
    light.shadowMapSize = CGSize(width: 4096, height: 4096)
    let directionalLightNode = SCNNode()
    directionalLightNode.eulerAngles = SCNVector3(x: GLKMathDegreesToRadians(-80.0), y: -
GLKMathDegreesToRadians(-180.0), z: 0.0)
    directionalLightNode.light = light
    self.sceneView.scene.rootNode.addChildNode(directionalLightNode)
  }



//MARK: PopUp presentation

    func presentPopUp(fromBottom: Bool) {
        var attributes = EKAttributes()

        attributes.position = .bottom
        attributes.entryBackground = .color(color: .white)
        attributes.displayDuration = .infinity
        attributes.entryInteraction = .absorbTouches
        attributes.screenInteraction = .forward
        attributes.positionConstraints.verticalOffset = 10
        attributes.roundCorners = .all(radius: 10)
        let widthConstraint = EKAttributes.PositionConstraints.Edge.ratio(value: 0.9)
        let heightConstraint = EKAttributes.PositionConstraints.Edge.intrinsic
        attributes.positionConstraints.size = .init(width: widthConstraint, height: heightConstraint)

        if !fromBottom {
            attributes.position = .top
        }

        SwiftEntryKit.display(entry: popUpView, using: attributes)

    }

    func presentNotification() {
        var attributes = EKAttributes()

        attributes.position = .top
        attributes.entryBackground = .color(color: .white)
        attributes.displayDuration = 5
        attributes.positionConstraints.verticalOffset = 10
        attributes.roundCorners = .all(radius: 10)
        let widthConstraint = EKAttributes.PositionConstraints.Edge.ratio(value: 0.9)
        let heightConstraint = EKAttributes.PositionConstraints.Edge.intrinsic
        attributes.positionConstraints.size = .init(width: widthConstraint, height: heightConstraint)


        SwiftEntryKit.display(entry: notificationView, using: attributes)
    }

    func presentNetworkNotification() {
        self.visualEffectView.isHidden = false
        self.visualEffectView.isUserInteractionEnabled = true
        self.view.insertSubview(self.guideView, aboveSubview: self.visualEffectView)
        self.guideView.center = self.visualEffectView.center
        self.guideView.transform = CGAffineTransform.init(scaleX: 1.3, y: 1.3)
        self.guideView.alpha = 0

        UIView.animate(withDuration: 0.5) {
            self.guideView.alpha = 1
            self.guideView.transform = CGAffineTransform.identity
        }
    }
```

```
  func removeNetworkNotification() {

      UIView.animate(withDuration: 0.3, animations: {
          self.guideView.transform = CGAffineTransform.init(scaleX: 1.3, y: 1.3)
          self.guideView.alpha = 0
          self.visualEffectView.isHidden = true
          self.visualEffectView.isUserInteractionEnabled = false
      }) { (success) in
          self.guideView.removeFromSuperview()
      }

  }

  @objc func handleThanksTap(_ sender: UITapGestureRecognizer) {
     print("Hello World")
      guideView.removeFromSuperview()
  }




  @IBAction func guideOKButtonPressed(_ sender: UIButton) {
  }



  @IBAction func popupCloseButtonPressed(_ sender: UIButton) {
      SwiftEntryKit.dismiss()
  }


  @IBAction func menuButtonPressed(_ sender: UIButton) {
      let menuVC = storyboard?.instantiateViewController(withIdentifier: "menuVC") as! OtherViewController
      present(menuVC, animated: true, completion: nil)
      menuVC.torchProjDelegate = self
  }

  //To update UI on the main thread
  func UI(_ block: @escaping ()->Void) {
      DispatchQueue.main.async(execute: block)
  }



// MARK: - ARSessionDelegate

func session(_ session: ARSession, didAdd anchors: [ARAnchor]) {
  guard let frame = session.currentFrame else { return }
  self.updateSessionInfoLabel(for: frame, trackingState: frame.camera.trackingState)
}

func session(_ session: ARSession, didRemove anchors: [ARAnchor]) {
  guard let frame = session.currentFrame else { return }
  self.updateSessionInfoLabel(for: frame, trackingState: frame.camera.trackingState)
}

func session(_ session: ARSession, cameraDidChangeTrackingState camera: ARCamera) {
  self.updateSessionInfoLabel(for: session.currentFrame!, trackingState: camera.trackingState)
}

// MARK: - ARSessionObserver
```

```
  func sessionWasInterrupted(_ session: ARSession) {
    // Inform the user that the session has been interrupted, for example, by presenting an overlay.
    self.sessionInfoLabel?.text = "AR сессія перервана"
  }

  func sessionInterruptionEnded(_ session: ARSession) {
    // Reset tracking and/or remove existing anchors if consistent tracking is required.
    self.sessionInfoLabel?.text = "AR сессія оновлена"
    // DID SOME SHIT
    //self.resetTracking()
  }

  func session(_ session: ARSession, didFailWithError error: Error) {
    self.sessionInfoLabel?.text = "Сталась помилка: \(error.localizedDescription)"
    guard error is ARError else { return }

    let errorWithInfo = error as NSError
    let messages = [
      errorWithInfo.localizedDescription,
      errorWithInfo.localizedFailureReason,
      errorWithInfo.localizedRecoverySuggestion
    ]

    // Remove optional error messages.
    let errorMessage = messages.compactMap { $0 }.joined(separator: "\n")

    DispatchQueue.main.async {
      // Present an alert informing about the error that has occurred.
      let alertController = UIAlertController(title: "Помилка при запуску AR сесії.", message: errorMessage,
preferredStyle: .alert)
      let restartAction = UIAlertAction(title: "Перезапустити", style: .default) { _ in
        alertController.dismiss(animated: true, completion: nil)
        self.resetTracking()
      }
      alertController.addAction(restartAction)
      self.present(alertController, animated: true, completion: nil)
    }
  }

  private func updateSessionInfoLabel(for frame: ARFrame, trackingState: ARCamera.TrackingState) {
    // Update the UI to provide feedback on the state of the AR experience.
    let message: String

    switch trackingState {
    case .normal where frame.anchors.isEmpty:
      // No planes detected; provide instructions for this app's AR interactions.
      message = "Найди табличку Початок Експозиции"

    case .notAvailable:
      message = "Трекінг недоступний."

    case .limited(.excessiveMotion):
      message = "Будь ласка, рухайте пристроєм повільніше."

    case .limited(.insufficientFeatures):
      message = "Трекінг обмежений - Наведіть пристрій на ділянку з видимою деталізацією поверхні."

    case .limited(.initializing):
      message = "Загрузка AR сесії."

    default:
      // No feedback needed when tracking is normal and planes are visible.
      // (Nor when in unreachable limited-tracking states.)
```

```
      message = self.projectAnchorManager != nil ? "Tap a plane to set project anchor" : ""
    }
    self.sessionInfoLabel?.text = message
  }

  private func resetTracking() {
    let configuration = ARWorldTrackingConfiguration()
    configuration.planeDetection = [.horizontal, .vertical]
    self.sceneView.session.run(configuration, options: [.resetTracking, .removeExistingAnchors])
  }

  class func emptyCoder() -> NSCoder {
    let data = NSMutableData()
    let archiver = NSKeyedArchiver(forWritingWith: data)
    archiver.finishEncoding()
    return NSKeyedUnarchiver(forReadingWith: data as Data)
  }

}

extension TorchProjectViewController: TorchProjectDelegate {
    func resetTorchProj() {
        loadedProject?.setScene(sceneName: "initial", resetCurrentScene: true)
        print("AR session started over")
    }
}
```

Б.8. PopUpContentManager.swift

```
import UIKit

var artObjects = [ArtObject]()

class PopUpContentManager {

    private init() {}
    static let shared = PopUpContentManager()

    //var artObjects = [ArtObject]()


    func loadData() {

        guard let url = URL(string:
"http://gsx2json.com/api?id=1f7j3WPBouLWuIhySO2p2bFF13RlR76BxvLSNFUT9Ods&columns=false") else {
            print("RETURNED HERE")
            return }

//http://gsx2json.com/api?id=11tvOkkWUcaYtmk5nVNOzMNx4v19VnhYuIqjC60zVrSw&sheet=1&columns=false
        URLSession.shared.dataTask(with: url) { (data, response
                , error) in
                guard let data = data else {
                  print("RETURNED")
                  return }

                do {
                   let decoder = JSONDecoder()
                  let rows = try decoder.decode(RootClass.self, from: data)
                  guard let JSONArt = rows.rows else { return }
```

```
                artObjects = JSONArt
                print("artObjects has \(artObjects.count) items")
            } catch let error {
                print("Error: ", error)
            }
        }.resume()

    }

}
```

Б.9 ArtObject.swift

```
import Foundation

struct RootClass : Codable {

    let rows : [ArtObject]?

    enum CodingKeys: String, CodingKey {
        case rows = "rows"
    }

    init(from decoder: Decoder) throws {
        let values = try decoder.container(keyedBy: CodingKeys.self)
        rows = try values.decodeIfPresent([ArtObject].self, forKey: .rows)
    }

}

struct ArtObject : Codable {

    let id : Int?
    let image : String?
    let name : String?
    let authoryear : String?
    let nextconnection : String?
    let trigger1description : String?
    let trigger1name : String?
    let trigger2description : String?
    let trigger2name : String?
    let trigger3description : String?
    let trigger3name : String?
    let trigger4description : String?
    let trigger4name : String?
    let trigger5description : String?
    let trigger5name : String?

    enum CodingKeys: String, CodingKey {
        case id = "id"
        case image = "image"
        case name = "name"
        case authoryear = "authoryear"
        case nextconnection = "nextconnection"
        case trigger1description = "trigger1description"
        case trigger1name = "trigger1name"
        case trigger2description = "trigger2description"
        case trigger2name = "trigger2name"
        case trigger3description = "trigger3description"
```

```
        case trigger3name = "trigger3name"
        case trigger4description = "trigger4description"
        case trigger4name = "trigger4name"
        case trigger5description = "trigger5description"
        case trigger5name = "trigger5name"
    }

    init(from decoder: Decoder) throws {
        let values = try decoder.container(keyedBy: CodingKeys.self)
        id = try values.decodeIfPresent(Int.self, forKey: .id)
        image = try values.decodeIfPresent(String.self, forKey: .image)
        name = try values.decodeIfPresent(String.self, forKey: .name)
        authoryear = try values.decodeIfPresent(String.self, forKey: .authoryear)
        nextconnection = try values.decodeIfPresent(String.self, forKey: .nextconnection)
        trigger1description = try values.decodeIfPresent(String.self, forKey: .trigger1description)
        trigger1name = try values.decodeIfPresent(String.self, forKey: .trigger1name)
        trigger2description = try values.decodeIfPresent(String.self, forKey: .trigger2description)
        trigger2name = try values.decodeIfPresent(String.self, forKey: .trigger2name)
        trigger3description = try values.decodeIfPresent(String.self, forKey: .trigger3description)
        trigger3name = try values.decodeIfPresent(String.self, forKey: .trigger3name)
        trigger4description = try values.decodeIfPresent(String.self, forKey: .trigger4description)
        trigger4name = try values.decodeIfPresent(String.self, forKey: .trigger4name)
        trigger5description = try values.decodeIfPresent(String.self, forKey: .trigger5description)
        trigger5name = try values.decodeIfPresent(String.self, forKey: .trigger5name)
    }

}
```