НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

«До захисту допущено»

Завідувач кафедри
_____ В.О. РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
“___” червня 2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою “Системне програмування та спеціалізовані комп’ютерні системи”

зі спеціальності **123 «Комп'ютерна інженерія»**

на тему: Система визначення лінійних та нелінійних відстаней пристроями з ОС iOS

Виконав : студент IV курсу, групи КВ-91
(шифр групи)

Батура Віталій Вікторович _____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник доц. каф. СПСКС, к.т.н. Боярінова Ю.Є _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

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

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2023 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»

Освітньо-професійна програма «Системне програмування та спеціалізовані комп’ютерні системи»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ В.О. РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
«___» червня 2023 р.

ЗАВДАННЯ
на дипломний проєкт студента
Батури Віталія Вікторовича

1. Тема проєкту «Система визначення лінійних та нелінійних відстаней пристроями з ОС iOS»,
керівник проєкту доц. каф. СПСКС, к.т.н. Боярінова Юлія Євгенівна,
затверджені наказом по університету від «31» травня 2023 р. №_2107-С

2. Термін подання студентом проєкту _____

3. Вихідні дані до проєкту: див. Технічне завдання.

4. Зміст пояснювальної записки
- Аналіз існуючих рішень
- Розробка додатку
- Тестування

5. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій тощо)
- Зв’язок структур, класів та фреймворків. Схема структурна;
- Визначення стану відстеження. Схема алгоритму;

- Відображення стану та отримання 3D-координати точки торкання. Схема алгоритму;
- Визначення дистанції та площі та відображення фігур на екрані. Схема алгоритму;
- Презентація за темою роботи.

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М., к.т.н., доцент каф. СПСКС		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Видача завдання на дипломне проектування	05.03.2023	
2.	Вивчення літератури за тематикою роботи	12.03.2023	
3.	Розроблення та узгодження технічного завдання	19.03.2023	
4.	Розроблення структури додатку	23.04.2023	
5.	Розроблення дизайну та графічних елементів	30.04.2023	
6.	Програмна реалізація додатку	07.05.2023	
7.	Тестування додатку	14.05.2023	
8.	Підготовка матеріалів текстової частини проекту	21.05.2023	
9.	Підготовка матеріалів графічної частини проекту	28.05.2023	
10.	Попередній огляд матеріалів диплому	01.06.2023	

Студент _____ Віталій БАТУРА
(підпис)

Керівник проєкту _____ Юлія БОЯРІНОВА
(підпис)

АНОТАЦІЯ

Бакалаврський дипломний проєкт включає пояснювальну записку (XX стор., YY рис., список використаної літератури з VV найменувань, 4 додатки).

Метою дипломного проекту є розробка додатку для пристроїв на базі операційної системи iOS, який надає можливість визначати лінійні та нелінійні відстані з використанням технології доповненої реальності ARKit. Проєкт був реалізований у відомому інтегрованому середовищі розробки Xcode, використовуючи мову програмування Swift.

Додаток, розроблений у рамках цього проекту, надає користувачам зручний інтерфейс для вимірювання довжини відрізку, довжини кола, півкола, а також площі кола і півкола. Завдяки використанню ARKit, користувачі можуть спрямовувати камеру свого iOS-пристрою на фізичні об'єкти у реальному світі та отримувати точні виміри цих об'єктів на екрані.

Розроблений додаток відкриває широкі перспективи в області вимірювання фізичних параметрів за допомогою мобільних пристроїв з iOS. Він може знайти застосування в різних галузях, включаючи будівництво, дизайн і архітектуру, де точні виміри є важливими чинниками.

Ключові слова:

СИСТЕМА ВИЗНАЧЕННЯ ВІДСТАНЕЙ, ARKIT, IOS, ДОПОВНЕНА РЕАЛЬНІСТЬ, XCODE, SWIFT.

ABSTRACT

The bachelor's diploma project comprises an explanatory report (XX pages, YY figures, a reference list of VV sources, and 4 appendices).

The objective of the project is to develop an application for iOS devices that enables the measurement of linear and non-linear distances using augmented reality technology ARKit. The project was implemented in the renowned integrated development environment Xcode, utilizing the Swift programming language.

The application developed as part of this project offers users a user-friendly interface for measuring the length of a segment, circumference, semicircle length, as well as the area of a circle and semicircle. By leveraging ARKit, users can direct the camera of their iOS device towards physical objects in the real world and obtain precise measurements of these objects on the screen.

The developed application opens up extensive possibilities in the domain of measuring physical parameters through iOS mobile devices. It can find applications across various fields, including construction, design, and architecture, where accurate measurements play a vital role.

Keywords:

DISTANCE MEASUREMENT SYSTEM, ARKIT, IOS, AUGMENTED REALITY, XCODE, SWIFT.

Пояснювальна записка
до дипломного проєкту

на тему: Система визначення лінійних та нелінійних відстаней пристроями з ОС iOS

Київ – 2023 року

ЗМІСТ

ДОДАТКИ

Додаток А. Копії графічних матеріалів

ІАЛЦ.045440.005 Д1 Зв'язок структур, класів та фреймворків. Схема структурна.

ІАЛЦ.045440.006 Д2 Визначення стану відстеження. Схема алгоритму.

ІАЛЦ.045440.007 Д3 Відображення стану та отримання 3D-координати точки торкання. Схема алгоритму.

ІАЛЦ.045440.008 Д4 Визначення дистанції та площі та відображення фігур на екрані. Схема алгоритму.

Додаток Б. Презентація

Додаток В. Довідка про впровадження

Додаток Г. Текст програмного коду

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

AR - це скорочення від "доповнена реальність" (англ. Augmented Reality). Це технологія, яка поєднує віртуальний світ з реальним оточенням, створюючи інтерактивний досвід для користувача.

ARKit - це фреймворк розробки доповненої реальності (AR) від Apple для мобільних пристроїв на базі iOS, таких як iPhone і iPad.

iOS - це мобільна операційна система, розроблена компанією Apple Inc. Вона призначена для використання на пристроях Apple, зокрема на iPhone, iPad і iPod Touch.

Xcode - це інтегроване середовище розробки (IDE) від Apple для розробки програмного забезпечення для платформ Apple, зокрема iOS, macOS, watchOS і tvOS.

SceneKit - це фреймворк для розробки тривимірної графіки, розроблений компанією Apple.

SCNNode – один з основних класів у фреймворку SceneKit. Він представляє об'єкт у тривимірній сцені та має властивості і методи для його розміщення, обертання, масштабування, анімації та багато іншого.

SCNVector3 - структура у фреймворку SceneKit, яка представляє тривимірний вектор з трьох компонентів: X, Y та Z.

ВСТУП

У сучасному світі, де технології займають центральне місце в багатьох аспектах життя, системи визначення відстаней є важливими інструментами для різноманітних застосувань. Вони забезпечують здатність вимірювати лінійні та нелінійні відстані між об'єктами і є основою для багатьох технологічних розробок.

З розвитком мобільних пристроїв, таких як смартфони та планшети з операційною системою iOS, вимоги до систем визначення відстаней стають все вищими. Користувачі сподіваються мати зручні та точні інструменти для вимірювання відстаней у реальному часі. Це стає особливо важливим у таких галузях, як архітектура, будівництво, дизайн та інші, де точне вимірювання відстаней має вирішальне значення для успішного виконання проектів.

Метою даного дипломного проекту є розробка системи визначення лінійних та нелінійних відстаней за допомогою пристроїв з операційною системою iOS. Використання технології розширеної реальності та сценаріїв використання AR-середовища дозволить створити зручний та ефективний інструмент для вимірювання відстаней.

Проект базується на використанні ARKit та SceneKit, двох потужних фреймворків розробки програмного забезпечення для створення реалістичних AR-додатків. ARKit надає розширені можливості визначення положення та сприйняття навколишнього середовища, а SceneKit дозволяє відтворювати тривимірні об'єкти та створювати візуальні ефекти.

В рамках дослідження та розробки, було створено мобільний додаток, який використовує функціонал ARKit та SceneKit для відображення віртуальних об'єктів у реальному середовищі з використанням камери пристрою. Додаток надає користувачу можливість вимірювати відстані між об'єктами, використовуючи взаємодію з фізичним середовищем та візуалізацію результатів на екрані пристрою.

Основною метою дослідження є розробка точного та ефективного алгоритму визначення відстаней, який дозволяє забезпечити якісні результати в реальному часі. Мета полягає в створенні додатку, який буде забезпечувати точність та надійність вимірювання відстаней у реальному часі, щоб задовольнити потреби користувачів та сприяти розвитку додатків на основі AR в різних сферах діяльності.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ

1.1. Доповнена реальність і її застосування у різних сферах

На даний момент технологія доповненої реальності існує вже доволі довго, тому і її використання є досить великим. AR з'явилась вперше у 1990-х роках, але набула популярності та широкого використання лише в останні роки. Але що ж таке доповнена реальність і в чому її різниця з віртуальною?

Віртуальна реальність – це комп'ютерна технологія, що створює віртуальне середовище, яке користувач може сприймати і взаємодіяти за допомогою спеціальних пристроїв, таких як VR-окуляри або гарнітура [1]. У віртуальній реальності користувач повністю занурюється у віртуальне середовище, яке може бути створене комп'ютерною графікою або за допомогою зйомки реального оточення.

AR - це технологія, що розширює фізичний світ за допомогою цифрових даних у режимі реального часу [2]. Доповнена реальність є складовою змішаної реальності і поєднує реальний світ з віртуальним, дозволяючи накладати віртуальні об'єкти, такі як графіка, звуки, анімація на оточуюче середовище. Тобто віртуальна реальність створює цілковито новий штучний світ, тоді як доповнена реальність додає окремі штучні елементи до сприйняття реального світу [3].

AR технологія розвивається дедалі більш швидкими темпами і знаходить все більше застосувань у різних сферах. Вона змінює спосіб, яким ми сприймаємо і взаємодіємо з навколишнім світом, доповнюючи його віртуальними об'єктами та інформацією.

Одним з ключових пристроїв для використання AR є смартфони. Принцип роботи технології AR полягає в наступному: камера пристрою AR захоплює зображення реальних об'єктів, а програмне забезпечення аналізує ці

зображення, поєднує їх з віртуальним контентом і виводить кінцеве зображення на екран пристрою.

Крім смартфонів, AR-окуляри є ще одним спеціальним пристроєм, який дозволяє використовувати AR. Вони надають користувачам можливість побачити віртуальні об'єкти прямо перед очима, створюючи враження, що вони знаходяться у реальному світі. AR-окуляри також можуть мати вбудовані камери та датчики, які допомагають відстежувати рухи користувача та взаємодіяти з навколишнім середовищем.

Застосування AR є досить різноманітними. Вона використовується у галузі розваг, дозволяючи користувачам грати в ігри, де віртуальні об'єкти і персонажі з'являються у реальному світі [4]. Компанії також використовують AR для реклами, створюючи інтерактивні презентації та віртуальні демонстрації своїх товарів і послуг.

AR також знайшла застосування у навчанні та тренуванні. Вона дозволяє створювати симуляції та віртуальні навчальні середовища, де студенти можуть отримати практичні навички у безпечних умовах. У медицині AR використовується для покращення хірургічних процедур та навчання майбутніх лікарів.

Доповнена реальність також знаходить своє застосування у сфері дизайну та архітектури. Вона дозволяє архітекторам та дизайнерам створювати віртуальні моделі будівель, інтер'єрів та об'єктів, що допомагає їм візуалізувати проекти та робити зміни до них в реальному часі. Це дозволяє замовникам та клієнтам отримати краще розуміння проекту та зробити більш обгрунтовані рішення.

У сфері туризму та подорожей AR може бути використана для створення підсиленої реальності під час екскурсій. Туристи можуть використовувати AR-пристрої або мобільні додатки, щоб отримати додаткову інформацію про історичні пам'ятки, визначні місця та культурні артефакти, коли вони

подорожують. Це дозволяє збагачувати їхній досвід подорожей та надавати додаткові контекстуальні дані.

AR також знаходить застосування у медіа і розважальній індустрії. Вона дозволяє створювати інтерактивні виставки, виступи та концерти, де віртуальні об'єкти можуть взаємодіяти з реальними артистами та виконавцями. AR також може бути використана для створення ігор, де реальний світ переплітається з віртуальним, надаючи гравцям захоплюючий досвід.

Наприклад, AR-гра "Pokémon Go" стала великим хітом, де гравці можуть ловити віртуальних покемонів в реальному світі, використовуючи свої смартфони [5]. Це є одним з прикладів того, як AR може змішувати віртуальні та реальні елементи для створення нових форм розваги.

Саме через розмаїття способів використання і високу продуктивність AR і надалі використовується і знаходить все більше і більше застосування.

1.2. Аналіз існуючих програм для роботи з AR

Давайте розглянемо наявні програмні рішення, які використовуються для взаємодії мобільних пристроїв з технологією доповненої реальності.

1.2.1. ARKit

Одним із широко використовуваних засобів для створення доповненої реальності є ARKit (рис. 1.1). Це фреймворк розробки, розроблений компанією Apple, який дозволяє розробникам створювати додатки доповненої реальності для пристроїв на базі iOS, таких як iPhone і iPad [6]. ARKit надає набір інструментів та можливостей, що спрощують розробку AR-додатків та забезпечують високу продуктивність та точність.

Рисунок 1.1 – набір інструментів ARKit

ARKit використовує вбудовану камеру, сенсори руху та обробку зображень на пристрої, щоб визначити положення та орієнтацію пристрою в просторі [7]. Він також надає функції для відстеження поверхонь, розпізнавання облич, вимірювання відстаней та інші можливості, які допомагають розробникам створювати різноманітні AR-додатки.

ARKit забезпечує такі функціональні можливості:

- Відстеження положення та орієнтації: ARKit дозволяє визначати точне положення та орієнтацію пристрою в просторі, що дозволяє точно розміщувати віртуальні об'єкти у реальному світі.
- Відстеження поверхонь: ARKit дозволяє виявляти та відстежувати поверхні, такі як столи, підлоги або стіни, для розміщення віртуальних об'єктів на них.
- Розпізнавання облич: ARKit має функціонал для розпізнавання та відстеження облич, що дозволяє створювати AR-додатки з функціями розпізнавання осіб або віртуальними масками.
- Вимірювання відстаней: ARKit дозволяє вимірювати відстані в реальному світі, що може бути корисно для розміщення об'єктів або виконання точних вимірювань.

ARKit створений з метою полегшити розробку AR-додатків для пристроїв на базі iOS, надаючи потужні інструменти та можливості, які дозволяють

розробникам творчо використовувати доповнену реальність для створення інноваційних додатків у різних галузях, включаючи ігри, освіту, дизайн та бізнес.

1.2.2. ARCore

ARCore - це платформа, розроблена Google, для створення AR-додатків, які взаємодіють з реальним світом (рис. 1.2). Основними функціями ARCore є відстеження руху і відображення віртуального контенту з точною прив'язкою до реального середовища [8].

Рисунок 1.2 - набір для розробки програмного забезпечення ARCore

ARCore використовує методи паралельної одометрії та відображення (COM) для відстеження руху пристрою у просторі. Він аналізує зображення з камери та визначає функціональні точки, які використовуються для обчислення зміни положення. Також враховуються дані з інерційного вимірювального пристрою (IMU), що допомагає визначити позицію та орієнтацію камери в просторі. Це дозволяє ARCore точно відстежувати рух пристрою і створювати віртуальну камеру, яка відображає 3D-контент з вірною перспективою.

Завдяки вирівнюванню віртуальної камери з позицією реальної камери пристрою, розробники можуть створювати AR-додатки, які накладають віртуальний контент на реальне зображення. Це дозволяє об'єднати віртуальний та реальний світ, створюючи вражаючі AR-сцени, де віртуальні об'єкти взаємодіють з реальними об'єктами у просторі.

ARCore виконує аналіз навколишнього середовища, шукаючи кластери характерних точок на горизонтальних і вертикальних поверхнях, таких як столи або стіни. Ці поверхні стають доступними для користувача для використання як площини віртуальної реальності. ARCore також може розпізнати межі кожної площини і передавати цю інформацію додатку користувача. За допомогою цієї інформації користувач може розміщувати віртуальні об'єкти на рівних поверхнях.

Оскільки ARCore використовує ознаки точок для аналізу площин, можуть виникати проблеми з виявленням плоских поверхонь без текстури, наприклад, білої стіни. Також ARCore може мати складнощі з виявленням площин при недостатньому освітленні.

ARCore може змінювати позиції якорів, оскільки воно постійно вдосконалює аналіз власної позиції та оточення. Якщо користувач бажає розмістити віртуальний об'єкт, він повинен визначити якорі, щоб ARCore міг відстежувати позицію об'єкта з часом. Зазвичай користувач створює якор з позиції дотику, який отримується під час хіт-тесту.

ARCore також може оновлювати положення об'єктів навколишнього середовища, таких як площини та 3D-об'єкти. Площини та точки є особливим типом відстежуваних об'єктів. Користувач може прив'язати віртуальні об'єкти до конкретних трекерів, щоб забезпечити стабільну взаємодію між віртуальним об'єктом і відстежуваним навіть при русі пристрою. Це означає, що якщо користувач розмістить віртуальний об'єкт на своєму столі, ARCore буде утримувати позицію пов'язаної площини, зберігаючи 3D-об'єкт на верхній поверхні столу.

Додатки доповненої реальності можуть використовувати доповнені зображення, щоб взаємодіяти з конкретними 2D-зображеннями, наприклад, упаковкою продукту або плакатом фільму. Користувачі можуть спрямовувати камеру свого телефону на ці зображення, і додаток буде накладати додаткову інформацію на них, що дозволяє взаємодіяти з доповненою реальністю.

Наприклад, користувач може сканувати візитку, і на екрані з'явиться додаткова детальна інформація про особу.

Зображення можуть бути прив'язані до певної інформації офлайн, а також можна створити базу даних зображень, до якої можна додавати об'єкти в реальному часі з пристрою. Після реєстрації ARCore виявлятиме ці зображення, їх межі та повертатиме відповідну позицію для накладання віртуальних об'єктів.

1.2.3. Vuforia

Vuforia є бібліотекою для розробки програмного забезпечення для мобільних пристроїв, яка дозволяє створювати додатки з використанням доповненої реальності (рис. 1.3). Вона використовує технологію комп'ютерного зору для розпізнавання та відстеження планарних зображень (графічних об'єктів) та простих 3D-об'єктів, таких як коробки, в режимі реального часу [9].

Рисунок 1.3 - набір для розробки програмного забезпечення доповненої реальності для мобільних пристроїв Vuforia

Можливість реєстрації зображень дозволяє розробникам позиціонувати та орієнтувати віртуальні об'єкти, такі як 3D-моделі, відносно зображень реального світу, коли їх переглядають через камеру мобільного пристрою. Віртуальний об'єкт потім відстежує положення та орієнтацію зображення в

режимі реального часу, забезпечуючи відповідну перспективу для спостерігача, що відповідає перспективі зображення маркера. Таким чином, створюється враження, що віртуальний об'єкт є частиною реальної сцени.

SDK Vuforia має підтримку різноманітних типів цілей, включаючи безмаркерні зображення, тривимірні 3D-конфігурації та VuMark - спеціальний тип маркера, який має форму адреси. У SDK також є додаткові можливості, такі як локалізоване виявлення оклюзії за допомогою віртуальних кнопок, вибір цільових зображень та можливість створення та перенастроювання маркерів програмно під час виконання.

Vuforia надає API для розробки додатків на мовах програмування C++, Java, Objective-C++ (комбінація синтаксису C++ та Objective-C) і .NET з використанням розширення для рушія Unity. Це означає, що SDK підтримує як нативну розробку для iOS і Android, так і розробку AR-додатків в Unity, які можна легко перенести на обидві платформи. AR-додатки, створені з використанням Vuforia, сумісні з різними мобільними пристроями, включаючи iPhone, iPad та Android-пристрої з ОС Android версії 2.2 або новішої, а також процесори ARMv6 або 7 з підтримкою Floating Point Unit (FPU).

Бібліотека Vuforia може використовуватися для розпізнавання та відстеження різних зображень та об'єктів [10], таких як:

- Зображення маркерів: це додавання вмісту на плоскі зображення, наприклад, друковані матеріали та упаковки продуктів.
- Багатоцільові маркери: ці маркери складаються з кількох зображень та можуть бути розміщені в звичайних геометричних фігурах (наприклад, коробках) або в будь-якому довільному розташуванні на плоских поверхнях.
- Маркерні моделі: це розпізнавання об'єктів за їх формою, використовуючи 3D-моделі. Це дає можливість додавати вміст AR на різноманітних предметах, таких як промислові пристрої, транспортні засоби, іграшки та побутові прилади.

- Об'єкти-маркери: ці маркери створюються шляхом сканування самого об'єкта. Вони добре підходять для використання в іграшках та інших виробах з детальною поверхнею та стабільною формою.
- Маркери для циліндрів: це зображення, розміщене на об'єктах, які мають наближену до циліндричної форму, наприклад, пляшки для напоїв, чашки для кави, банки з содою.
- Ground Plane: ця функція дозволяє розмістити вміст на горизонтальних поверхнях, таких як столи та підлоги.
- VuMarks: це спеціально налаштовані маркери, які можуть кодувати різні формати даних. Вони забезпечують унікальну ідентифікацію та відстеження для AR-додатків.
- Extended tracking: ця особлива функція дозволяє бачити об'єкт AR навіть після втрати розпізнаваного маркера. Розширене відстеження використовує IMU пристрій для покращення точності відстеження та збереження його, навіть коли ціль виходить з області видимості камери.
- Зовнішня камера: доступ до відеоданих з камери за межами мобільного пристрою, коли створюється доповнена реальність.

1.3. Аналіз існуючих додатків

Оскільки під час розробки проекту мав доступний смартфон від компанії Apple, було вирішено використовувати функціональні можливості бібліотеки ARKit для взаємодії з доповненою реальністю.

У цьому підрозділі розглянемо існуючі додатки, створені використовуючи ARKit. Так як даний фреймворк існує вже доволі довгий проміжок часу, то і програм є велика кількість.

1.3.1. Мірило

В просторах інтернету існує багато застосунків які знаходять відстані за допомогою ARKit. Одним із найрозповсюдженіших є стандартний додаток iPhone "Мірило", який ми можемо бачити на рис. 1.4. Цей додаток використовує вбудовану функціональність ARKit для надання можливості користувачам вимірювати розміри об'єктів у реальному світі за допомогою камери свого пристрою [11].

Рисунок 1.4 – додаток "Мірило"

За допомогою програми "Мірило" користувач може легко виміряти довжину об'єктів, наведений на екран пристрою. Для цього достатньо просто спрямувати камеру на дві крайні точки об'єкта, а додаток автоматично визначить відстань між ними та покаже результат на екрані, що ми можемо бачити на рис. 1.5. Крім того, додаток "Мірило" надає можливість робити фотографії з вимірами та зберігати їх для подальшого використання.

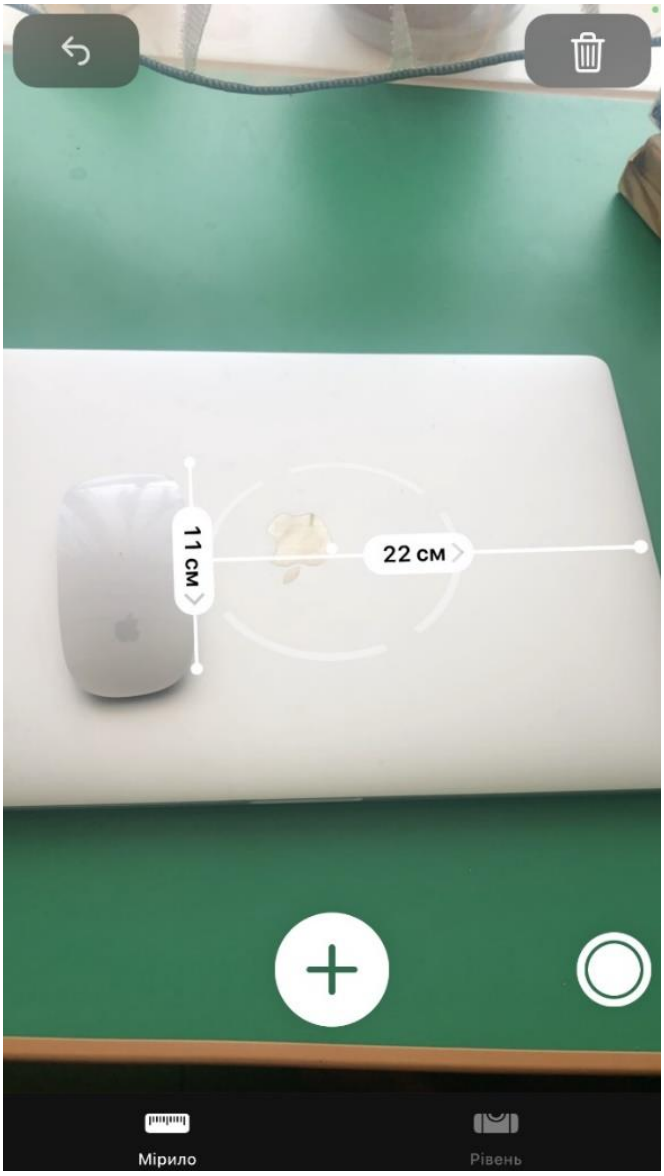

Рисунок 1.5 – визначення лінійних відстаней за допомогою додатка "Мірило"

Недоліком даного додатка є неможливість визначення нелінійних відстаней та площі фігур. Мірило використовує прямі лінії для вимірювання відстаней між двома точками в просторі. Це означає, що додаток не може точно виміряти відстані та площі фігур, якщо вони мають складні круглі форми.

Отже, хоча додаток "Мірило" є корисним інструментом для швидкого вимірювання прямих відстаней у просторі, він має свої обмеження і не може забезпечити вимірювання нелінійних відстаней та площі фігур.

1.3.2. MeasureKit – AR Ruler Tape

Іншим доволі розповсюдженим додатком для вимірювання відстаней за допомогою технології доповненої реальності є "MeasureKit – AR Ruler Tape", який ми можемо бачити на рис. 1.6.

Рисунок 1.6 – додаток "MeasureKit – AR Ruler Tape"

Цей додаток має більший спектр можливостей, які ми можемо бачити на рис. 1.7, порівняно з додатком "Мірило". Додаток "MeasureKit" дозволяє вимірювати не тільки прямі відстані між двома точками, але і нелінійні відстані, такі як довжини кривих ліній або шляхів [12]. Також доступні функції визначення кутів, висоти людини.

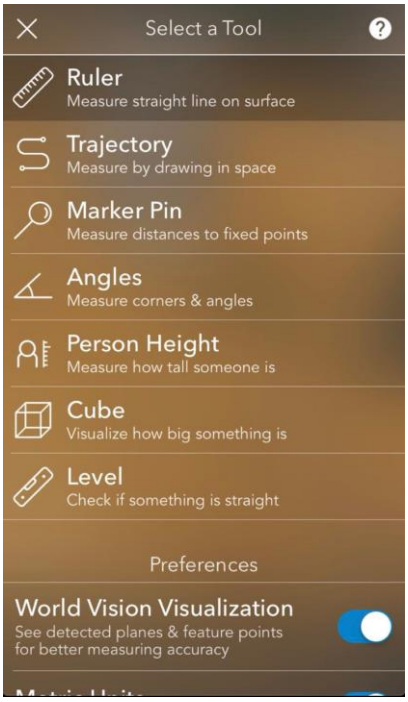

Рисунок 1.7 – функції мобільного додатка "MeasureKit – AR Ruler Tape"

"MeasureKit" має інтуїтивний і зручний інтерфейс, який дозволяє легко встановлювати точки вимірювання та отримувати результати. Додаток також підтримує функцію збереження вимірювань та обміну результатами з іншими користувачами через різні канали комунікації.

Але і цей застосунок, хоч і має багато можливостей, не має змоги побудови кола і визначення його довжини і площі.

1.3.3. AirMeasure – AR Tape & Ruler

Ще одним розповсюдженим додатком є "AirMeasure – AR Tape & Ruler", який ми можемо бачити на рис. 1.8. Програма також має можливість вимірювання лінійних відстаней, за допомогою двох точок у фізичному просторі.

Рисунок 1.8 – додаток "AirMeasure – AR Tape & Ruler"

Програма має широкий функціонал, що ви можете бачити на рис. 1.9. Додаток має функції вимірювання розмірів прямокутних об'єктів, таких як вимірювання довжини, ширини і висоти прямокутних об'єктів, наприклад столів, картин на стіні або меблів [13]. Ще можна визначити прямі кути і рівень. Це може бути корисно для точного розташування меблів або будівництва. Додаток "AirMeasure - AR Tape & Ruler" також має функцію "Virtual Furniture" (віртуальні меблі). Ця функція дозволяє вам переглянути, як будь-які меблі або предмети інтер'єру будуть виглядати в вашому просторі за допомогою доповненої реальності. Ще програма дає можливість малювання чи накладання креативних ефектів на фотографії або відео за допомогою доповненої реальності. Також як і в інші розглянутих додатках ви можете зберігати результати вимірювань для подальшого використання або надсилати їх іншим людям через електронну пошту або повідомлення.

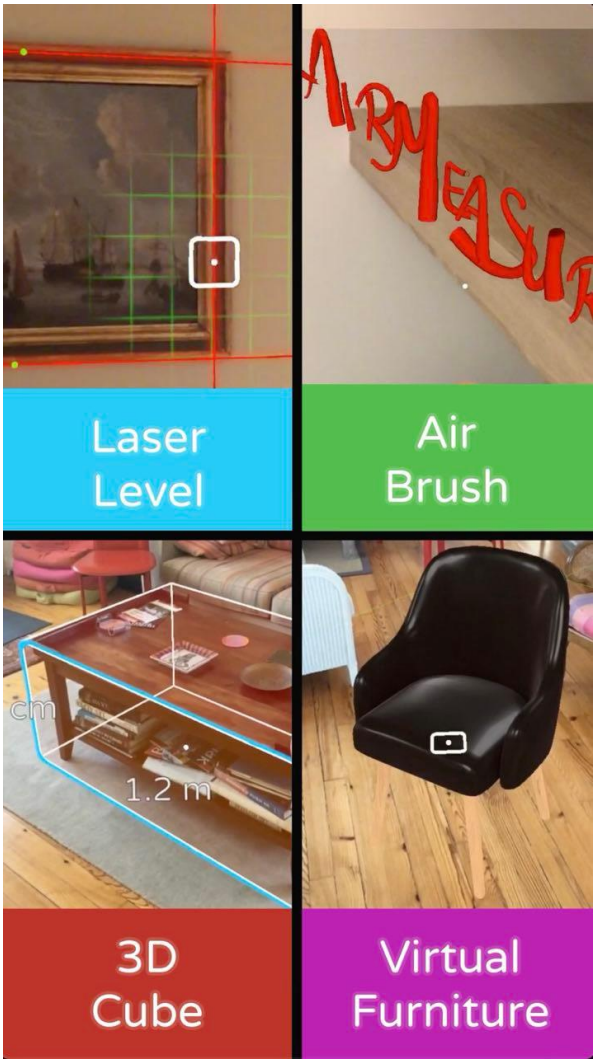

Рисунок 1.9 – функціонал додатку "AirMeasure – AR Tape & Ruler"

Недоліком "AirMeasure - AR Tape & Ruler" є обмежена функціональність щодо вимірювання нелінійних фігур та об'єктів, зокрема кіл. Це означає, що ви не зможете точно виміряти довжину кола або обчислити його площу без використання додаткових засобів.

1.4. Обґрунтування теми дипломного проекту

Отже, з аналізу існуючих додатків можна зробити висновок, що існує попит на програму, яка б здатна визначати в реальному часі довжину кола, півкола і площу цих геометричних фігур за допомогою доповненої реальності. Цей додаток може бути корисним для широкого спектра користувачів, включаючи професіоналів і приватних осіб.

Для професіоналів, таких як архітектори, дизайнери, які займаються проектуванням будівель, додаток з визначенням геометричних параметрів кола та півкола може бути цінним інструментом. Він може допомогти їм виміряти різні геометричні параметри, такі як довжина стіни, радіус або площа колони. Це спростить процес проектування та планування простору, дозволяючи швидко і точно виміряти необхідні параметри.

У галузі інженерії, зокрема при проектуванні доріг, тунелів або інших інфраструктурних об'єктів, вимірювання площі та довжини кола може бути також важливим. Додаток, що використовує ARKit, дозволяє точно визначати ці параметри в реальному часі, що спрощує роботу інженерів та забезпечує точність результатів.

Ще додатком залюбки будуть користуватися садівники та ландшафтні дизайнери. Він може допомогти визначити площу газону, круглого квітника або півкола пагорба. Це дозволить точно розрахувати необхідну кількість матеріалів, наприклад, трави або рослин, для озеленення даної площі.

Додаток може бути корисним у навчанні учні та студентів геометрії та математики. Він надає можливість студентам вивчати та експериментувати з геометричними формами, демонструючи відношення між діаметром, радіусом, площею та довжиною кола.

Також такий додаток зацікавить широке коло користувачів, включаючи приватних осіб. Навіть звичайні люди можуть скористатись його функціоналом під час виконання побутових завдань. Наприклад, вони зможуть виміряти довжину як круглого, так і прямокутного столу, визначити розміри газону у своєму саду або оцінити величину приміщення для правильного розміру меблів чи матеріалів. Завдяки точним обрахункам, ці люди зможуть зробити правильний вибір і забезпечити оптимальне використання простору та ресурсів.

Отже, розробка додатку, який може визначати довжину кола, півкола і площу за допомогою доповненої реальності, може задовольнити потреби різних груп користувачів і мати практичне застосування в різних сферах діяльності.

2. РОЗРОБКА СИСТЕМИ ВИМІРЮВАННЯ ЗА ДОПОМОГОЮ ARKIT

2.1. Середовище розробки Xcode

Для розробки додатку вирішив скористатись інтегрованим середовищем розробки (IDE) від Apple Xcode (рис. 2.1).

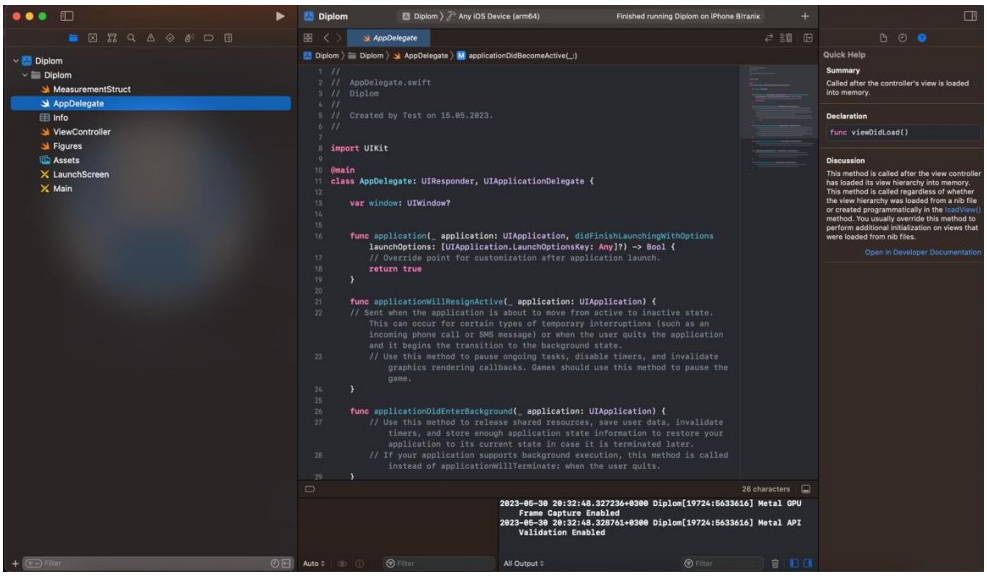

Рисунок 2.1 - інтегрованим середовищем розробки Xcode

Xcode - це інтегроване середовище розробки від Apple для створення програмного забезпечення для різних платформ, зокрема iOS, macOS, watchOS та tvOS [14]. Воно є основним інструментом для розробки програмного забезпечення для пристроїв Apple. Хоча Xcode є і основним інструментом розробки для платформ Apple, існують деякі альтернативи, які також можуть бути використані на Mac для розробки програмного забезпечення. Проте важливо зазначити, що альтернативи мають свої переваги і недоліки порівняно з Xcode, і вибір залежатиме від вашого конкретного випадку використання. Та я вважаю Xcode найкращим і найзручнішим середовищем розробки. Ось кілька альтернатив Xcode:

- Visual Studio Code (рис. 2.2): Visual Studio Code (VS Code) є безкоштовним текстовим редактором, розробленим Microsoft [15]. Він підтримує різні мови програмування, включаючи Swift і Objective-C,

та надає широкий спектр розширень, що розширюють його функціональність. Він має потужну систему редактора коду, автодоповнення, вбудовані інструменти для керування версіями та підтримку відлагодження. Однак він не має такої інтегрованої підтримки для розробки iOS-додатків, як Xcode, і вимагає додаткових кроків для налаштування розробки на платформі Apple.

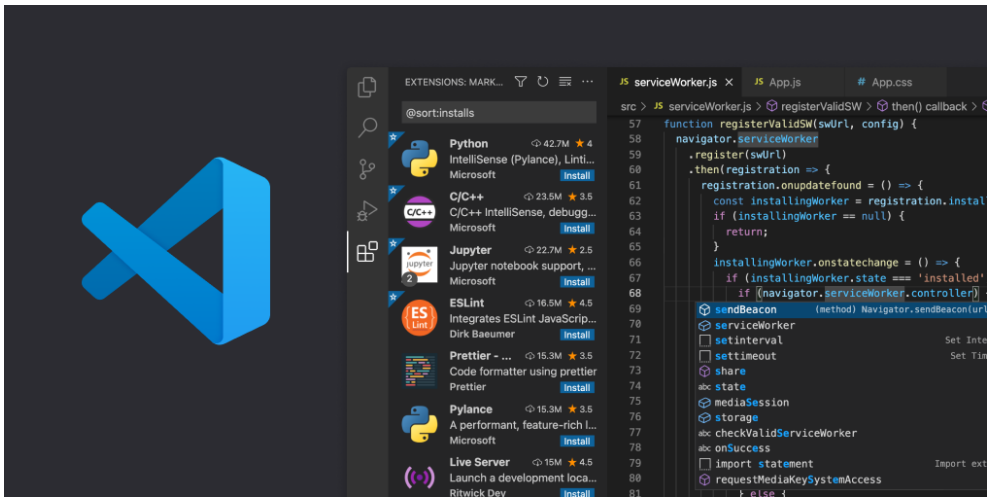

Рисунок 2.2 - Visual Studio Code

- AppCode (рис. 2.3): AppCode є інтегрованою середовищем розробки (IDE) від компанії JetBrains, яка спеціалізується на інструментах розробки програмного забезпечення [16]. AppCode спеціально створений для розробки додатків для платформ Apple з використанням Swift, Objective-C і C/C++. Він має підтримку для розширення функціональності, редактор коду з автодоповненням, рефакторингом та вбудованими інструментами для відлагодження. Однак AppCode не надає таку повну інтеграцію з екосистемою Apple, як Xcode, і може бути менш популярним серед розробників.

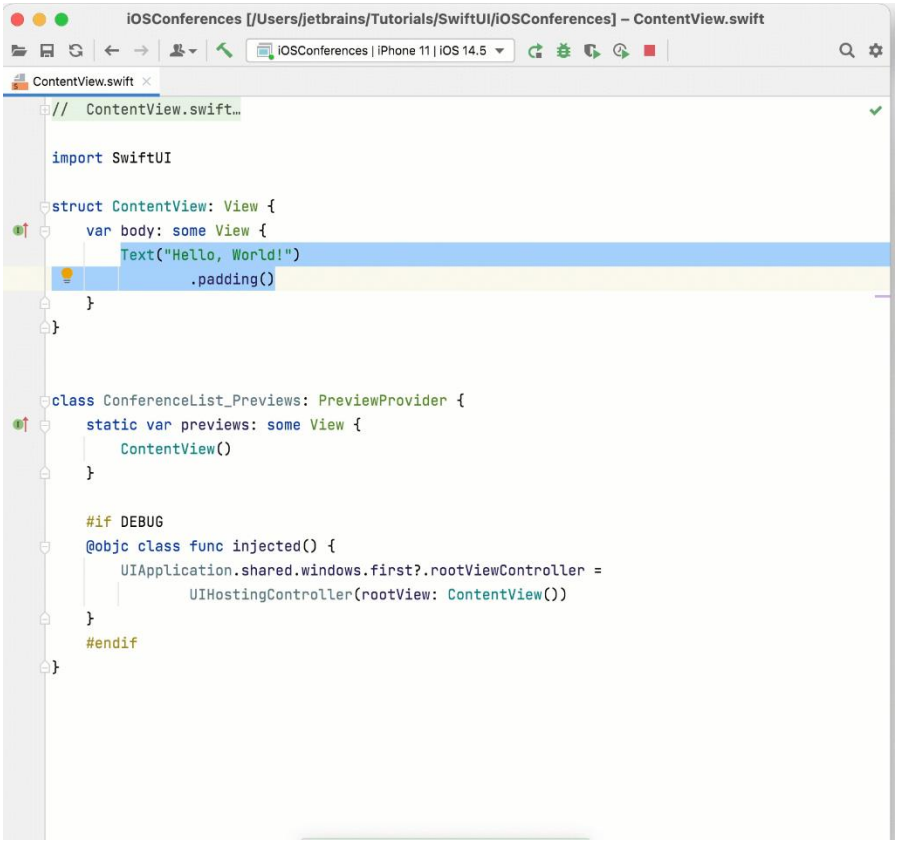

Рисунок 2.3 - AppCode

- IntelliJ IDEA (рис. 2.4): IntelliJ IDEA є іншим IDE від JetBrains, яке підтримує різні мови програмування, включаючи Swift та Objective-C. Воно має потужні можливості редактора коду, автодоповнення, рефакторинг, відлагодження та інші інструменти розробки [17]. Хоча IntelliJ IDEA може бути зручним для розробки програмного забезпечення, воно не має такої спеціалізації на розробку для платформ Apple, як Xcode, і може вимагати додаткових налаштувань для роботи з проектами iOS або macOS.

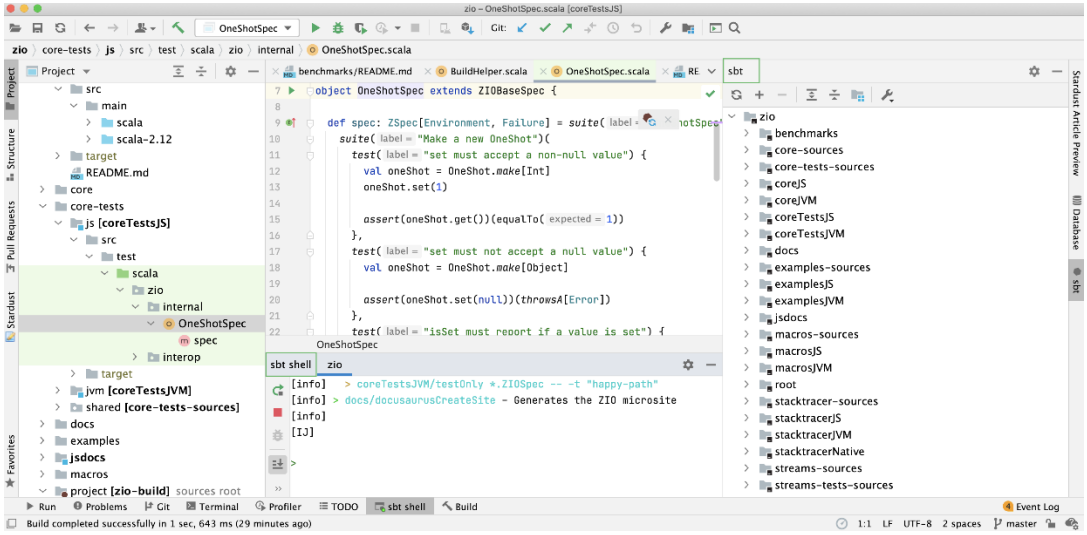

Рисунок 2.4 - IntelliJ IDEA

Отже, ми бачимо, що Xcode має переваги над альтернативами і надає повну інтеграцію з екосистемою Apple, включаючи доступ до інструментів, таких як Interface Builder для розробки інтерфейсу користувача, підтримку відлагодження на реальних пристроях та інструменти для розгортання додатків в App Store. Він також має багато ресурсів і документації, які спрощують розробку для платформ Apple. Тому, для розробки додатків для iOS, macOS, watchOS або tvOS, Xcode є найбільш оптимальним вибором.

Ось деякі переваги та функції Xcode:

- Інтегроване середовище розробки: Xcode надає повний набір інструментів, які потрібні для розробки програмного забезпечення для платформ Apple. Це включає в себе редактор коду, компілятор, дебагер, інструменти розробки інтерфейсу користувача та інші.
- Підтримка мов програмування: Xcode підтримує кілька мов програмування, включаючи Swift, Objective-C та AppleScript. Swift є основною мовою для розробки програмного забезпечення для пристроїв Apple, і Xcode надає потужні інструменти для розробки на цій мові.
- Інструменти для інтерфейсу користувача: Xcode має інструменти, що спрощують розробку інтерфейсу користувача, такі як Interface

Builder. Ви можете створювати і налаштовувати вигляд і поведінку вашого додатка за допомогою інтуїтивно зрозумілих інструментів перетягування та розміщення.

- Дебагер та профілювання: Xcode має потужні засоби для дебагу та профілювання вашого коду. Ви можете встановлювати точки зупинки, відстежувати значення змінних, аналізувати профілі продуктивності та багато іншого.
- Симулятори та відлагодження на реальних пристроях: Xcode постачається зі вбудованими симуляторами, що дозволяють вам випробувати ваше програмне забезпечення на різних пристроях та операційних системах без необхідності володіння фізичними пристроями. Ви також можете відлагоджувати та тестувати ваше програмне забезпечення на реальних пристроях, підключених до вашого комп'ютера.
- Інтеграція з іншими інструментами Apple: Xcode підтримує інтеграцію з іншими інструментами та сервісами Apple, такими як Instruments для аналізу продуктивності, TestFlight для бета-тестування, утиліти пакетування та розгортання додатків та багато іншого.
- Багатофункціональність: Xcode має багато інших корисних функцій, таких як система керування версіями, інтеграція зі сторонніми бібліотеками та фреймворками, вбудована документація та підтримка локалізації.

Тому для розробки будемо використовувати інтегроване середовище розробки (IDE) від Apple Xcode.

2.2. Створення проекту

Створивши новий проект, одразу додамо необхідні фреймворки ARKit і SceneKit.

```
import UIKit
import SceneKit
import ARKit
```

У вашому проекті використовується ARKit для створення доповненої реальності додатку. Основними функціями, які нам будуть потрібні з ARKit, будуть відображення об'єктів AR та детекція поверхонь. З використанням ARKit ми створюємо та відображаємо віртуальні об'єкти у реальному світі. Наприклад, нам треба буде вивести круг і лінію, які представляють виміряні об'єкти. В цьому нам допоможе використовуєте ARKit. ARKit також допоможе виявляти горизонтальні поверхні, такі як підлогу або стіл, на які можна розміщувати віртуальні об'єкти. Це дозволяє нам використовувати ці поверхні як основу для розміщення та вимірювання об'єктів.

Також буде корисно визначати стан відстеження камери. Це допомагає користувачеві знати, коли відстеження об'єктів у реальному світі є достатнім для вимірювань.

Також ми будемо взаємодіяти з вимірювальними об'єктами, наприклад, встановлювати початкову та кінцеву точки вимірювання та отримувати відповідні значення відстані та площі.

SceneKit є високорівневою бібліотекою для рендерингу 3D-графіки, яка надає потужні можливості для створення і відображення 3D-сцен у програмах для iOS і macOS [18]. Вона використовується для створення візуально захоплюючих ігор, симуляторів, віртуальних туристичних додатків, а також додатків з розширеною реальністю.

Основні можливості SceneKit включають:

- Віртуальна сцена: SceneKit дозволяє створювати віртуальні сцени зі складними 3D-об'єктами, освітленням, камерами та ефектами. Ви

можете створювати сцени з одним або кількома 3D-об'єктами та розміщувати їх у віртуальному просторі.

- 3D-об'єкти: Ви можете створювати різноманітні 3D-об'єкти, такі як геометричні форми, моделі персонажів, трансформовані об'єкти та багато іншого. SceneKit підтримує різноманітні типи геометрії, включаючи меші, примітиви, скульптурні моделі та імпортовані моделі з форматів файлів, таких як Collada або Alembic.
- Анімація: SceneKit надає потужні можливості анімації 3D-об'єктів. Ви можете створювати рухи, трансформації, зміну кольору, зміну прозорості та багато іншого. Анімації можна виконувати як вручну, так і за допомогою ключових кадрів або складних систем анімації.
- Освітлення: SceneKit підтримує різні типи освітлення, такі як точкове освітлення, напрямлене освітлення та реалістичне освітлення з джерелами світла. Ви можете контролювати параметри освітлення для створення різних ефектів освітлення у ваших сценах.
- Фізика: SceneKit включає фізичний двигун, який дозволяє симулювати фізичну поведінку об'єктів. Ви можете задати масу, геометрію, матеріал та фізичні властивості для об'єктів, щоб вони взаємодіяли один з одним у віртуальному просторі.
- Камери: Ви можете додавати камери до сцени і керувати їх положенням та орієнтацією. Камери дозволяють змінювати точку огляду сцени та керувати ракурсом, полем зору і іншими параметрами огляду.
- Ефекти: SceneKit підтримує різні візуальні ефекти, такі як тіні, розмивання, відбиття, прозорість, рельєфність та багато інших. Ви можете додавати ці ефекти до об'єктів або до всієї сцени для створення більш реалістичного відображення.

- Інтеграція з ARKit: SceneKit має підтримку ARKit, що дозволяє створювати додатки з розширеною реальністю. Ви можете поєднувати сцени SceneKit з функціями ARKit, такими як відстежування положення та розмірів об'єктів у реальному світі, відображення віртуальних об'єктів на реальних поверхнях та іншими AR-функціями, що якраз нам дуже підходить.

З функцій бачимо, що для нашого додатку можна використовувати SceneKit для створення і відображення 3D-моделей об'єктів у віртуальному середовищі. Наприклад, ми можемо створювати лінію і круг за допомогою SceneKit і додавати їх до сцени AR.

Далі створимо клас і додамо протоколи ARSCNViewDelegate та UIViewController. ARSCNViewDelegate є протоколом делегата для ARSCNView, який відповідає за обробку подій та взаємодію з AR сценою. UIViewController є основним класом в UIKit, який використовується для управління користувацьким інтерфейсом на iOS пристроях. Він виконує роль контролера вікна, керуючи відображенням та взаємодією з іншими видами, які входять у його компоненти.

Обов’язково треба додати можливість використання камери, щоб перед відкриттям програми питався дозвіл на її застосування. Для цього створюємо Property List файл, в якому вибираємо можливість використовувати камеру і підписуємо опис, який буде з’являтися при запиті чи використовувати камеру для даного додатку. Це можна побачити на рис. 2.3.

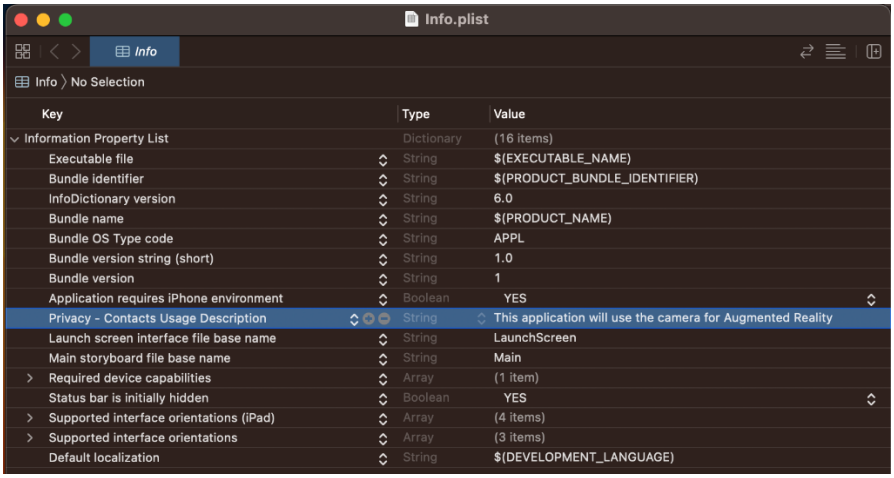

Рисунок 2.3 – файл списку властивостей Info.plist

Також корегуємо інші властивості та переходимо до графічної побудови нашої системи визначення відстаней.

2.3. Графічна побудова додатку

Для визначення відстаней ми будемо використовувати значення координат, розміщених у просторі, які будуть визначатися по середній точці нашого смартфона. Тому, для зручності знаходження потрібної нам точки (початку виміру і кінця), давайте поставимо в середині екрана позначку “+”, яку назвемо Aim. Для більшого привертання до неї уваги, зробимо її червоною (рис. 2.5).

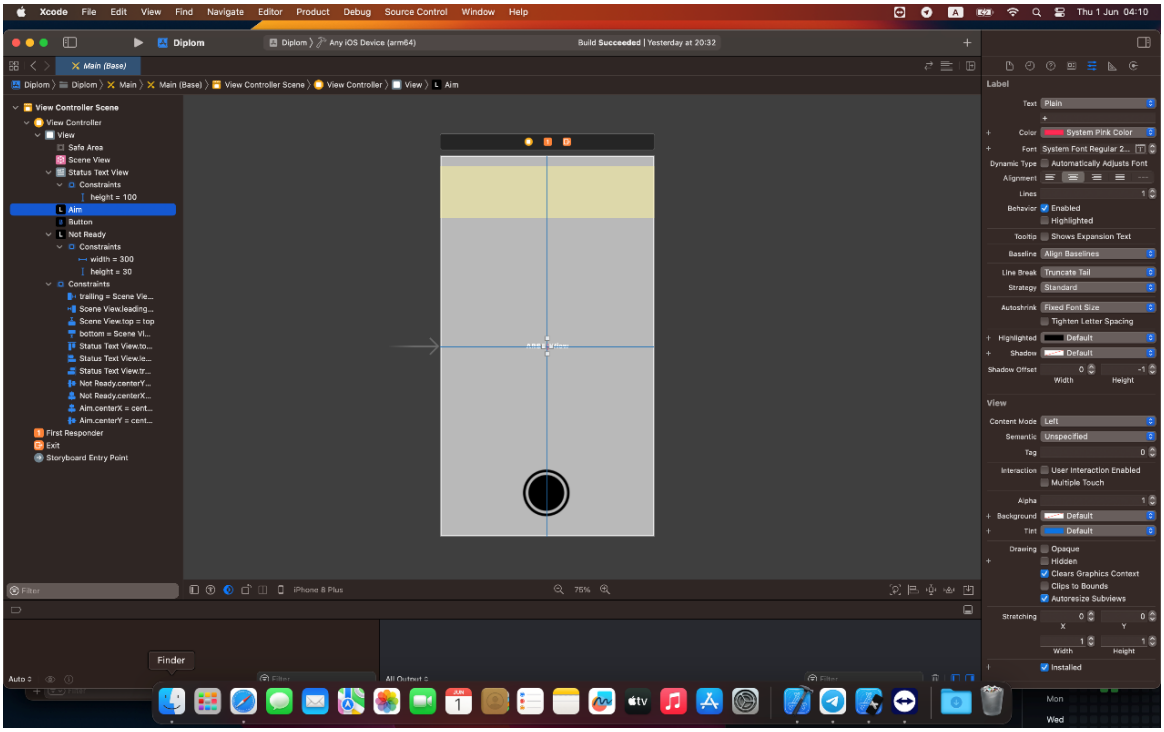

Рисунок 2.5 – позначка Aim в центрі екрану

Також потрібно зробити кнопку, за допомогою якої ми будемо визначати початок виміру і кінець. Для того щоб кнопка мала гарний вигляд треба знайти картинку, яка б нам сподобалась, що і зробимо. Картинку кнопки додамо до папки Assets.xcassets в створену папку button_circle.imageset. Також в цій папці додамо в файл Contents.json назву нашої картинки для розміру 1x. Тепер ми можемо використовувати картинку для нашої кнопки (рис. 2.6).

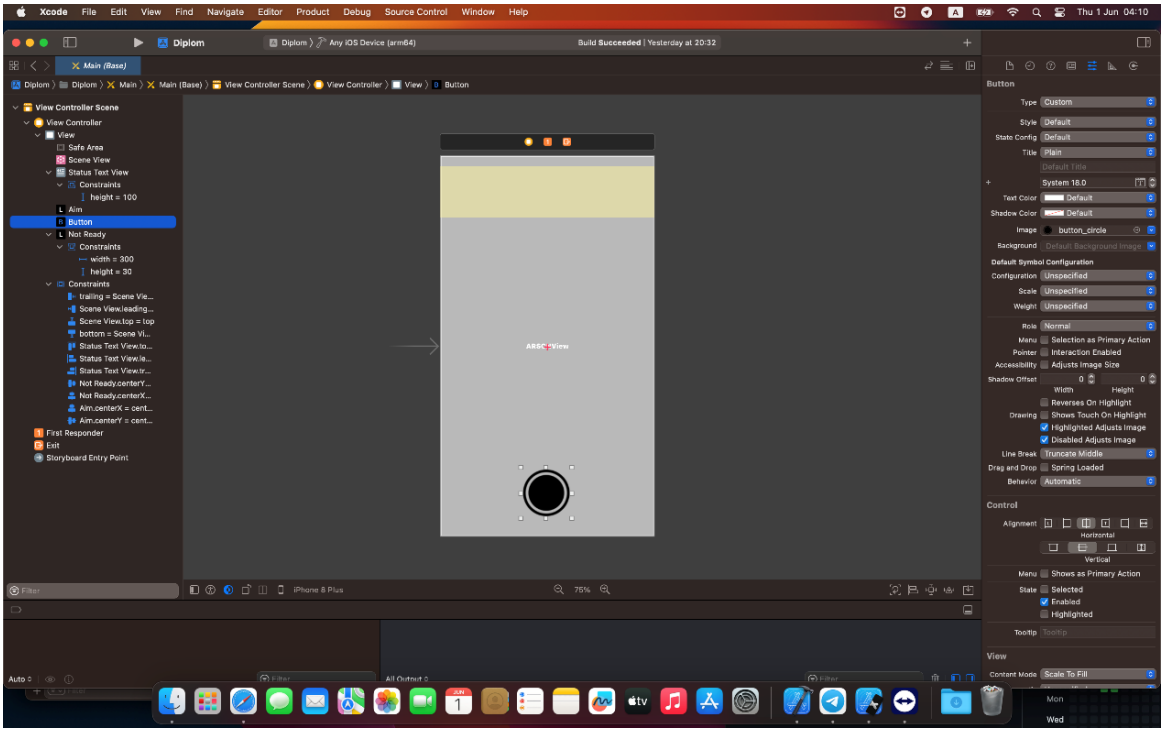

Рисунок 2.6 – кнопка Button

Для виведення результатів нам також потрібне текстове поле (рис. 2.5). Зробимо його 100 пікселів у висоту і в ширину на весь екран. Поставимо колір тексту червоний, вирівнювання по центру, текст з нахилом. Саме ж текстове поле зробимо прозоро-жовтим.

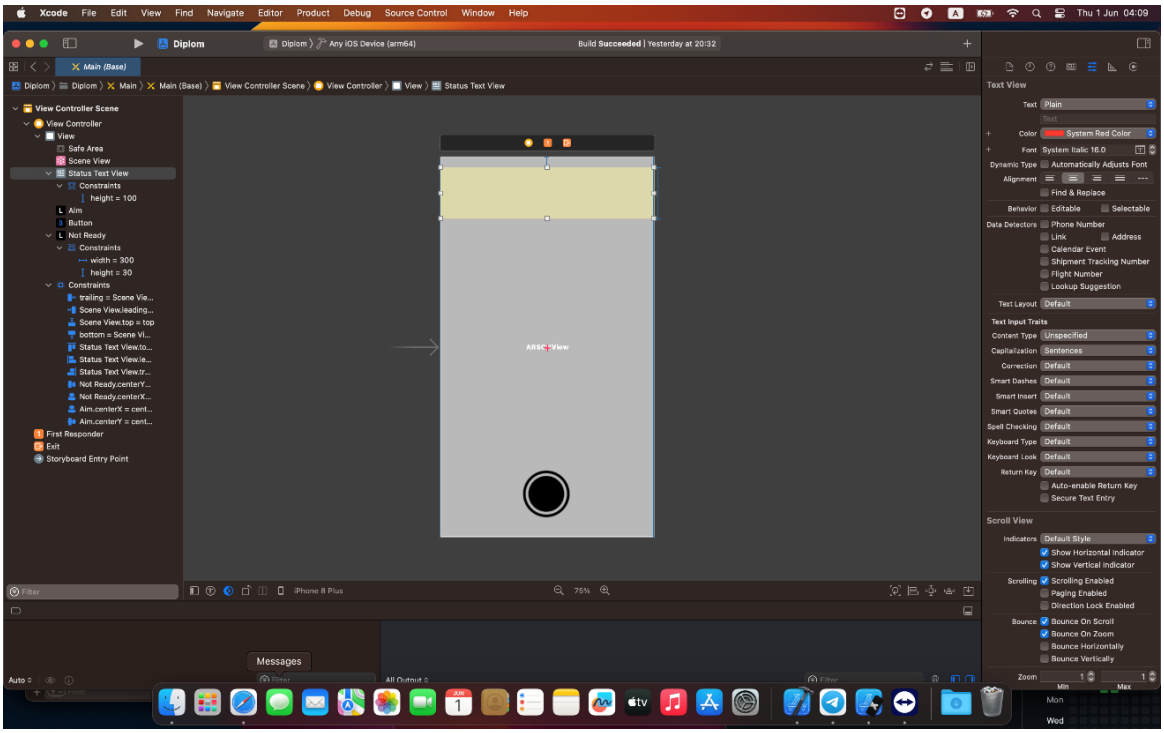

Рисунок 2.5 – текстове поле Status Text View

Ще зробимо позначку Not Ready для можливості виводити повідомлення про готовність до вимірювань, або неготовність через якісь обмеження вимірювань. Визначимо максимальну довжину і висоту та поставимо вирівнювання тексту по центру.

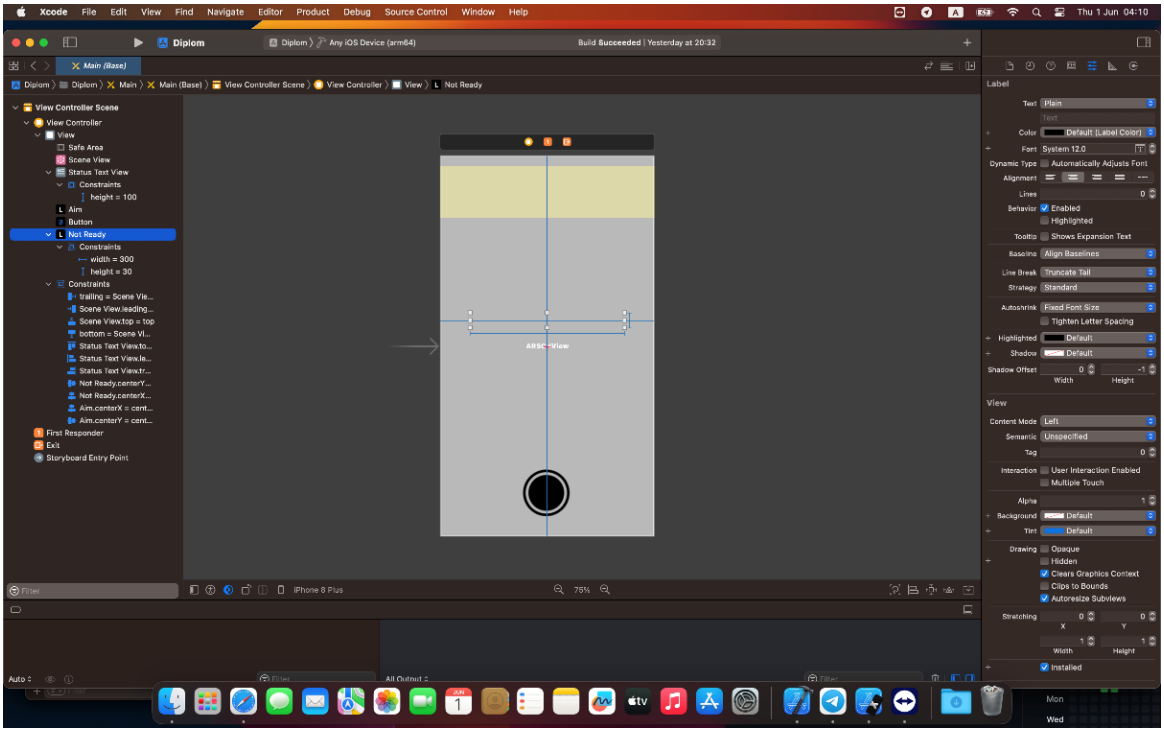

Рисунок 2.6 – позначка Not Ready

Тепер нам залишилось зробити обмеження наших елементів по розташуванню. Ось і готова графічна побудова додатку.

2.4. Програмна розробка додатку

Для початку слід підключити зроблені нами графічні елементи [19].

```
@IBOutlet weak var sceneView: ARSCNView!
@IBOutlet weak var statusTextView: UITextView!
@IBOutlet weak var aim: UILabel!
@IBOutlet weak var notReady: UILabel!
@IBOutlet weak var sceneView: ARSCNView!
```

А для кнопки button зробимо функцію, яка буде збільшувати лічильник нажимань на кнопку на одиницю. Для цього попередньо створимо змінну лічильника нажимань і зробимо йому початкове значення нуль.

```
@IBAction func Button(_ sender: Any) {
        touchesOnButtonCounter += 1
}
```

Разом з цим створимо змінні, які ми будемо використовувати далі, та встановимо їм початкові значення.

```
var isMeasuring = false
var activeStartingPoint: SCNVector3!
var cylinderNode = SCNNode()
var torusNode = SCNNode()
var touchesOnButtonCounter = 0
var seccount = 0

var trackingState: ARCamera.TrackingState!
```

Після цього створимо метод viewDidLoad(), який є перевизначеною функцією, яка викликається після завантаження виду до пам'яті. Далі викликаємо метод super.viewDidLoad(), щоб гарантувати виконання необхідного налаштування, визначеного у суперкласі.

Властивість delegate об'єкта sceneView встановлюємо на self, що означає, що поточний екземпляр виконуватиме роль делегата для sceneView. Шляхом відповідності протоколу ARSCNViewDelegate, ViewController може реагувати на події та надавати власну поведінку для AR-виду сцени.

```
override func viewDidLoad() {
    super.viewDidLoad()

    sceneView.delegate = self
}
```

Створимо метод viewWillAppear(_ animated: Bool), який також є перевизначеною функцією, яка викликається перед тим, як вид з'явиться на екрані. Викликаємо метод super.viewWillAppear(animated), щоб гарантувати виконання необхідного налаштування, визначеного у суперкласі.

Далі, створюємо конфігурацію сесії AR з визначенням площини. Конфігурація ARWorldTrackingConfiguration дозволяє відстежувати розташування та орієнтацію пристрою у світі AR. Встановлюємо площинну детекцію .horizontal, що означає, що будуть виявлятись горизонтальні площини.

Потім викликаємо метод run(_ configuration: ARConfiguration) сесії sceneView, передаючи створену конфігурацію. Це запускає AR-сесію та починає відстежування пристрою та детектування площин в реальному часі.

```
override func viewWillAppear(_ animated: Bool) {
    super.viewWillAppear(animated)

    let configuration = ARWorldTrackingConfiguration()
    configuration.planeDetection = .horizontal

    sceneView.session.run(configuration)
}
```

І створимо останню перевизначену функцію viewWillDisappear(_ animated: Bool), яка викликається перед тим, як вид зникне з екрану. Викликаємо метод super.viewWillDisappear(animated), щоб гарантувати виконання необхідних дій у суперкласі.

Далі, викликаємо метод pause() сесії sceneView. Цей метод призупиняє AR-сесію, зупиняючи відстежування розташування та орієнтації пристрою, а також детекцію площин.

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

    sceneView.session.pause()
}
```

Створимо функцію session(_ session: ARSession, cameraDidChangeTrackingState camera: ARCamera). Вона відповідає за обробку змін стану відстеження камери в AR-сесії. Основна задача цієї функції полягає у збереженні останнього відомого стану відстеження камери у змінну trackingState.

Кожного разу, коли стан відстеження камери змінюється, ця функція спрацьовує. Вона приймає два параметри: session, який представляє AR-сесію, і camera, який містить інформацію про поточну камеру.

У тілі функції trackingState присвоюється значення camera.trackingState, тобто поточний стан відстеження камери. За допомогою цього коду програма може отримати доступ до останнього стану відстеження камери і

використовувати його для подальшої обробки або відображення на інтерфейсі користувача.

```
func session(_ session: ARSession, cameraDidChangeTrackingState camera: ARCamera) {
    trackingState = camera.trackingState
}
```

Також потрібно створити функцію session(_ session: ARSession, didFailWithError error: Error). Вона викликається в разі виникнення помилки під час AR-сесії.

Ця функція має два параметри:

session: об'єкт ARSession, який відповідає за виконання AR-сесії, що не вдалася.

error: об'єкт типу Error, який містить інформацію про виниклу помилку.

```
func session(_ session: ARSession, didFailWithError error: Error) {
    //Present an error message to the user
}
```

Ще потрібно створити дві функції пов’язані з сесіює – це sessionWasInterrupted(_ session: ARSession) і sessionInterruptionEnded(_ session: ARSession). Перша викликається, коли AR-сесія була перервана, наприклад, через те, що інша програма взяла управління камерою пристрою або з'явилося вхідне сповіщення.

Ця функція має один параметр session: об'єкт ARSession, який був перерваний.

```
func sessionWasInterrupted(_ session: ARSession) {
    //Inform the user that the session has been interrupted, for example, by presenting an overlay
}
```

sessionInterruptionEnded(_ session: ARSession) же викликається, коли переривання AR-сесії завершилося і сесія може продовжитися.

Ця функція має також один параметр session: об'єкт ARSession, для якого було завершено переривання.

```
func sessionInterruptionEnded(_ session: ARSession) {
    //Reset tracking and/or remove existing anchors if consistent tracking is required
}
```

Тепер нам потрібно створити таку функцію, за допомогою якої буде виконуватись оновлення сцени на кожному кадрі. Тому ми створимо функцію renderer(_ renderer: SCNSceneRenderer, updateAtTime time: TimeInterval).

Ця функція має два параметри:

renderer: об'єкт SCNSceneRenderer, який відповідає за відображення сцени.

time: час в секундах, коли відбувається оновлення.

У реалізації даної функції використаємо чергу диспетчеризації DispatchQueue.main.async, щоб запустити основну програму, яку ми будемо створювати пізніше, на головній черзі диспетчеризації. Це забезпечує, що оновлення інтерфейсу користувача відбувається на головній черзі, оскільки багато методів SceneKit повинні виконуватись на головній черзі.

Наприклад, якщо у вас є функція prog(), яка виконує певну логіку оновлення сцени або інтерфейсу користувача, то виклик self.prog() в методі renderer(_:updateAtTime:) гарантує, що ця функція буде виконуватись на головній черзі диспетчеризації.

```
func renderer(_ renderer: SCNSceneRenderer, updateAtTime time: TimeInterval) {
    DispatchQueue.main.async {
        self.prog()
    }
}
```

Для того, щоб основна програма працювала тільки тоді, коли камера відстежує потрібно створити функцію getTrackigDescription(), яка визначає опис стану відстеження камери на основі значення trackingState. Отже, вона буде повертати рядок, який містить опис стану відстеження.

Спочатку оголошуємо змінну description як порожній рядок. Ця змінна буде використовуватись для збереження опису стану відстеження камери.

Функція перевіряє, чи існує значення trackingState за допомогою опціонального зв'язувального оператора if let. Якщо trackingState має значення, то виконується наступний код. Якщо trackingState є nil, тоді значення description залишається порожнім, і функція повертає порожній рядок.

Далі використовуємо оператор switch для перевірки значення trackingState і визначення конкретного стану відстеження.

У випадку, коли trackingState дорівнює .notAvailable, це означає, що відстеження недоступне. У такому випадку значенню description присвоюємо рядок "TRACKING UNAVAILABLE".

У випадку, коли trackingState дорівнює .normal, це означає, що відстеження в нормальному стані, тобто камера готова до вимірювання. У такому випадку значенню description присвоюємо рядок "READY TO MEASURE".

У випадку, коли trackingState має значення .limited(let reason), це означає, що відстеження обмежене і містить певну причину обмеження. Для визначення причини обмеження використовуємо ще один вкладений оператор switch зі значенням reason.

Якщо reason дорівнює .excessiveMotion, це означає, що камера занадто активно рухається. У такому випадку значенню description присвоюємо рядок "TRACKING LIMITED - Too much camera movement".

Якщо reason дорівнює .insufficientFeatures, це означає, що на поверхні недостатньо деталей для відстеження. У такому випадку значенню description присвоюємо рядок "TRACKING LIMITED - Not enough surface detail".

Якщо reason дорівнює .initializing, це означає, що відстеження знаходиться у процесі ініціалізації. У такому випадку значенню description присвоюємо рядок "INITIALIZING".

Якщо reason дорівнює .relocalizing, це означає, що відстеження обмежене, оскільки спробує відновити відстеження після переривання. У такому випадку значенню description присвоюємо рядок "TRACKING LIMITED - Attempting to resume after an interruption".

У іншому випадку, коли reason має значення, яке не відомо на момент написання коду, використовуємо вираз @unknown default. Це означає, що якщо

з'явиться іший варіант значення reason, то значенню description буде присвоєно рядок "TRACKING LIMITED - Unknown error".

Ця функція повертає значення description, яке містить опис поточного стану відстеження камери. Якщо trackingState було nil, то значення description залишається порожнім.

```
func getTrackigDescription() -> String {
    var description = ""
    if let t = trackingState {
      switch(t) {
      case .notAvailable:
        description = "TRACKING UNAVAILABLE"
      case .normal:
        description = "READY TO MEASURE"
      case .limited(let reason):
        switch reason {
        case .excessiveMotion:
          description = "TRACKING LIMITED - Too much camera movement"
        case .insufficientFeatures:
          description = "TRACKING LIMITED - Not enough surface detail"
        case .initializing:
          description = "INITIALIZING"
        case .relocalizing:
          description = "TRACKING LIMITED - Attempting to resume after an interruption"
        @unknown default:
          description = "TRACKING LIMITED - Unknown error"
        }
      }
    }
    return description
}
```

Також створимо структуру Measurement, яка представляє виміряні значення для доповненої реальності. Для початку ініціалізуємо потрібні нам значення:

```
let start: SCNVector3
let end: SCNVector3
let distance: Float
let vector: SCNVector3
let halfwayPoint: SCNVector3
let lengthOfCircle: Float
let lengthOfSemicircle: Float
let areaOfCircle: Float
let areaOfSemicircle: Float
```

SCNVector3 - це структура з бібліотеки ARKit, яка представляє тривимірний вектор у просторі SceneKit з координатами x, y і z. Вона використовується для визначення і роботи з точками, векторами і напрямками у тривимірному просторі.

Тепер створимо ініціалізатор init(start: SCNVector3, end: SCNVector3), який приймає початкову та кінцеву точки вимірювання. В ньому ми будемо обчислювати значення для всіх властивостей структури на основі цих точок.

start буде представляти початкову точку вимірювання у тривимірному просторі, а end представляє кінцеву точку вимірювання у тривимірному просторі. Ці координати будуть передаватись нам за допомогою камери і ARKit з реального світу.

За допомогою SCNVector3Make створюємо SCNVector3 vector, який є вектором між початковою та кінцевою точками.

Тепер, distance буде знаходити відстань цього вектору, тобто відстань між початковою та кінцевою точками вимірювання. Для цього ми будемо використовувати формулу Евклідової відстані:

$$\sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2},$$

де $p = (p_1, p_2, \ldots, p_n)$ – перша точка;

$q = (q_1, q_2, \ldots, q_n)$ – друга точка.

Тому, скориставшись даною формулою, запишемо її за допомогою коду.

```
self.distance = sqrtf(vector.x * vector.x + vector.y * vector.y + vector.z * vector.z)
```

Далі для розміщення фігур і розрахунків нам потрібна буде серединна точка між початковою та кінцевою точками вимірювання halfwayPoint. Тому, запишемо і цей код.

```
self.halfwayPoint = SCNVector3(
    start.x + vector.x / 2.0,
    start.y + vector.y / 2.0,
    start.z + vector.z / 2.0
)
```

Наша програма буде визначати довжину кола, отже ми будемо використовувати знайому всім формулу:

$$2 * \pi * r = \pi * d,$$

де r – радіус кола;

d – діаметр.

Відстань між початком вимірювання та кінцем і буде діаметром, тому використаємо другу з цих формул.

```
self.lengthOfCircle = distance * Float.pi
```

Відповідно довжиниа півкола lengthOfSemicircle буде дорівнювати значенню lengthOfCircle поділеному на два.

Для знаходження площі кола застосуємо формулу:

$$\pi * r^2 = \frac{1}{4} * \pi * d^2,$$

де r – радіус кола;

d – діаметр.

Оскільки у нас уже є діаметр, то скористаємось формулою з ним.

```
self.areaOfCircle = distance / 2.0 * distance / 2.0 * Float.pi
```

Тепер розпочнемо створення нашої основної функції prog(), сцена якої буде обновлюватись кожен кадр і ми будемо бачити виведення фігур і значень довжини відрізка, кола, півкола, а також площі кола і півкола. Отже, вона буде відповідати за керування режимом вимірювання та відображенням потрібних елементів на екрані.

Для початку зробимо перевірку, чи готова до відстеження камера. Тобто чи поточний стан відстеження getTrackigDescription() дорівнює "READY TO MEASURE". Це означає, що система ARKit готова до вимірювання і можна продовжити збирати дані для вимірювання.

Якщо стан відстеження є "READY TO MEASURE", тоді ми робимо поле statusTextView та aim видимими, встановлюючи isHidden в false. Це означає, що на екрані відображатимуться відповідні елементи.

Потім перевіряємо, чи змінна seccount менша за 100. Початкове значення цієї змінної дорівнює нулю. Це контролює, щоб позначка notReady, яка буде виводити на екрані повідомлення про готовність камери до вимірювання, була видима тільки протягом певного періоду часу і не закривала екран і його вміст. Для цього якщо seccount менше 100, то встановлюємо текст notReady на поточний стан відстеження, отриманий з функції getTrackigDescription(), а seccount збільшуємо на 1.

Коли seccount досягне 100, тобто достатньо часу минуло, поле notReady стає прихованим, встановлюючи isHidden в true.

```
statusTextView.isHidden = false
aim.isHidden = false
if seccount < 100 {
   notReady.text = getTrackigDescription()
   seccount += 1
} else {
   notReady.isHidden = true
}
```

Далі використаємо метод hitTest(_:types:). Він використовується для виконання перетину променя з вмістом сцени у додатку розширеної реальності. Цей метод приймає два параметри: point - точка на екрані, в якій ви бажаєте виконати перетин променя, і types - типи об'єктів, з якими ви бажаєте виконати перетин.

Нам потрібен центр нашого екрану, тому self.sceneView ми використовуємо для доступу до сцени AR, а self.sceneView.center визначає центр екрану. Таким чином, метод hitTest(_:types:) виконує перетин променя з точкою, яка знаходиться у центрі екрану.

featurePoint є одним з типів перетину, який може бути використаний в методі hitTest(_:types:). Він представляє собою точку особливості (feature point) у сцені AR.

Точки особливості або точки ознак (feature points) - це визначені особливості, які ARKit виявляє в реальному світі під час процесу відслідковування поверхні або структури сцени. Вони можуть представляти

розпізнавані деталі, такі як кути, розмітки або інші особливості візуального оточення.

Тому ми і використаємо .featurePoint у методі, що дозволить нам виконати перетин променя з цими точками особливості у сцені AR.

Результат виклику hitTest(_:types:) є масивом об'єктів типу ARHitTestResult. Кожен об'єкт ARHitTestResult містить інформацію про перетин променя з об'єктом у сцені AR. Далі ми викликаємо first в масиві результатів перетину, щоб отримати перший (найближчий) результат.

Також застосовуємо операцію "optional binding" (guard let) для перевірки, чи є результат перетину доступним (не є nil). Якщо результат недоступний, виконується вихід з функції за допомогою ключового слова return. Якщо результат доступний, він зберігається у змінній hitResult для подальшого використання у коді.

```
guard let hitResult = self.sceneView.hitTest(self.sceneView.center, types: .featurePoint).first else {
        return
}
```

Отже, даний метод дозволяє виконати перетин променя зі сценою AR у заданій точці на екрані та отримати результати перетину з об'єктами в сцені.

Тепер з отриманих даних hitResult отримуємо координати точки зіткнення point, які використовуються для вимірювання та відображення візуальних елементів на сцені.

```
let point = SCNVector3Make(hitResult.worldTransform.columns.3.x,
                           hitResult.worldTransform.columns.3.y,
                           hitResult.worldTransform.columns.3.z)
```

Далі будемо перевіряти чи користувач натиснув кнопку один раз, тобто чи змінна touchesOnButtonCounter дорівнює 1. Це означає, що користувач натиснув на кнопку для початку вимірювання.

Потім перевіримо чи ще не працює вимірювання, тобто змінна isMeasuring дорівнює false. Тоді ми запишемо початкову точку і збережемо її у змінній activeStartingPoint.

```
if isMeasuring == false {
        activeStartingPoint = point
}
```

Одразу після цієї перевірки змінимо isMeasuring на true, що означає, що вимірювання вже відбувається.

Далі створимо об'єкт measurement типу нашої структури Measurement, використовуючи для цього початкову точку activeStartingPoint та кінцеву точку point.

```
let measurement = Measurement(start: activeStartingPoint, end: point)
```

Тепер попередньо видалимо створені раніше візуальні елементи з сцни за допомогою функції deleteNode, яка приймає в якості параметру node - об'єкт типу SCNNode, який потрібно видалити з сцени.

У тілі функції викликається метод enumerateChildNodes на кореневому вузлі сцени sceneView.scene.rootNode. Цей метод перебирає всі дочірні вузли кореневого вузла та виконує передану замикання для кожного з них.

У замиканні, яке передається методу enumerateChildNodes, ми можемо виконувати деякі операції з кожним дочірнім вузлом. У нашому випадку, замикання приймає два параметри: node і stop. Кожен дочірній вузол передається як аргумент до цього замикання.

Тобто внутрішнє замикання просто видаляє кожен дочірній вузол з його батьківського вузла, використовуючи метод removeFromParentNode(). Це призводить до видалення і зі сцени.

Таким чином, функція deleteNode перебирає всі дочірні вузли кореневого вузла сцени і видаляє їх з сцени, що ефективно призводить до видалення всіх об'єктів з сцени.

```
deleteNode(cylinderNode)
deleteNode(torusNode)

func deleteNode(_ node: SCNNode) {
    sceneView.scene.rootNode.enumerateChildNodes { (node, stop) in
        node.removeFromParentNode()
    }
}
```

Тепер розпочнемо створення фігур і виведення їх на сцену. Нам потрібно, щоб було видно лінію – відстань та коло. Але для того щоб фігури не були

плоскими і, змінивши наше розташування або розташування камери, ми могли бачити фігури такими самими нам потрібно буде створювати лінію і коло в 3D. Тому ми будемо створювати циліндр і тор.

Циліндр - це геометрична фігура, яка має форму тіла обертання, складається з двох кругових основ та бічної поверхні, яка їх з'єднує [20]. Нам він ідеально підійде, якщо його “покласти на бік” (рис. 2.7), тобто відстань – це буде його висота. Циліндр буде краще паралелепіпеда тому, що, подивившись під різним кутом на нього, він завжди буде однакової товщини. Циліндр буде мати однаковий радіус і не мати ребер, як паралелепіпед.

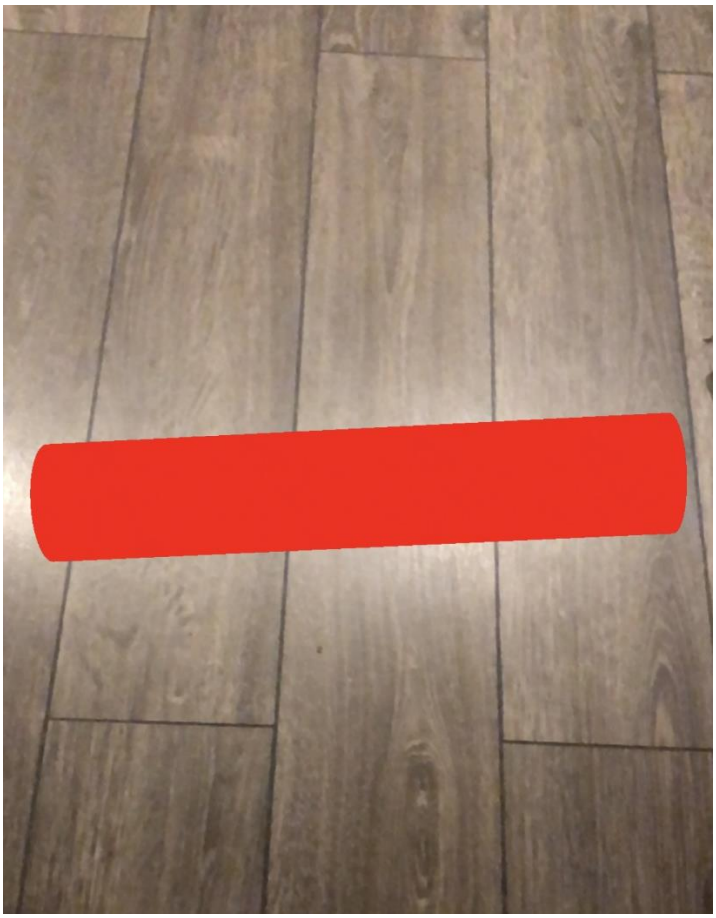

Рисунок 2.7 – приклад циліндра, створеного за допомогою SceneKit

По прикладу циліндру, в якому ми обертаємо багатокутника навколо відрізка, візьмемо тор. Тор (тороїд) - це геометрична фігура, яка може бути описана як поверхня, отримана обертанням кола навколо вісі, яка не лежить у площині цього кола [21] (рис. 2.8).

Рисунок 2.8 – приклад тора, створеного за допомогою SceneKit

Отже, для початку створимо новий вузол endNode, який розміщується у кінцевій точці point. Це нам буде потрібно для подальшого розміщення циліндра і тора в просторі, будемо використовувати цей вузол кінцевої точки як орієнтир, на який будуть орієнтуватись фігури.

Для початку створимо об'єкт cylinder, який представляє циліндр у сцені AR. Вказуємо його радіус рівний 0.002, що якраз добре видно і не закриває велику частину екрану, і висоту (CGFloat(measurement.distance)) - останнє значення взято з обчисленої властивості distance у структурі Measurement.

```
let cylinder = SCNCylinder(radius: 0.002, height: CGFloat(measurement.distance))
```

Тепер створимо вузол (SCNNode), який використовує створений циліндр (cylinder) як геометрію. Це дозволяє розмістити циліндр у сцені AR.

```
let cylinderNode = SCNNode(geometry: cylinder)
```

Далі встановимо позицію вузла циліндра на половині відрізка між початковою та кінцевою точками вимірювання. Це забезпечує коректне розташування циліндра відносно вимірювання.

```
cylinderNode.position = measurement.halfwayPoint
```

Тепер нам потрібно встановити орієнтацію вузла циліндра і його розміщення у просторі. Для цього ми можемо використати функцію look(at:up:localFront:). Дана функція в SceneKit визначає, як об'єкт повинен орієнтуватися у просторі.

Основні параметри функції:

at: Позиція, на яку циліндр повинен спрямовуватися. У нашому випадку, це позиція endNode.position, тобто циліндр буде спрямований в кінцеву точку.

up: Вектор, що вказує напрямок "вгору" для орієнтації об'єкта. Ми використаємо sceneView.scene.rootNode.worldUp, що визначає напрямок "вгору" відносно кореневого вузла сцени. Це допомагає забезпечити правильну орієнтацію об'єкта у відношенні до глобальної координатної системи.

localFront: Локальний вектор, що вказує напрямок "вперед" для орієнтації об'єкта. У нашому випадку, створимо SCNVector3Make(0, 1, 0), який вказує на напрямок "вгору" для циліндра, тобто циліндр буде спрямований вертикально.

```
cylinderNode.look(at: endNode.position, up: sceneView.scene.rootNode.worldUp, localFront: SCNVector3Make(0, 1, 0))
```

Цей рядок коду гарантує, що циліндр буде спрямований на endNode.position з відповідною орієнтацією у просторі і буде торкатися двох точок – початкової і кінцевої.

Тепер нам залишилось тільки додати вузол циліндра до кореневого вузла сцени AR (sceneView.scene.rootNode). Це зробить циліндр видимим у сцені AR.

```
sceneView.scene.rootNode.addChildNode(cylinderNode)
```

Аналогічно будемо створювати і тор. Спочатку створюємо об'єкт SCNTorus, який представляє тор у сцені AR. Вказуються його радіуси: радіус кільця (CGFloat(measurement.distance/2)) - половина відстані вимірювання, та радіус труби такий самий як радіус циліндра - 0.002.

```
let torus = SCNTorus(ringRadius: CGFloat(measurement.distance/2), pipeRadius: 0.002)
```

Після цього створюємо вузол, який використовує створений тор як геометрію. Також встановлюємо позицію вузла тору на половині відрізка між початковою та кінцевою точками вимірювання.

```
let torusNode = SCNNode(geometry: torus)
torusNode.position = measurement.halfwayPoint
```

Для встановлення орієнтації створимо вектор directionToViewer, який визначає напрямок завжди перпендикулярний до позиції камери. За допомогою цього тор буде будуватись на площині, що видна і перпендикулярна до користувача, що, в свою чергу, дозволить бачити коло.

```
let directionToViewer = SCNVector3(
        cameraPosition.x - measurement.halfwayPoint.x,
        cameraPosition.y - measurement.halfwayPoint.y,
        cameraPosition.z - measurement.halfwayPoint.z
)
```

Тепер встановлюється орієнтація вузла тору за допомогою тієї ж функції look(at:up:localFront:), щоб він спрямовувався на кінцеву точку вимірювання. Вектор, що спрямовується вгору, визначається як directionToViewer. Локальний фронт встановлюється як вектор [1, 0, 0], що означає, що тор спрямований горизонтально у бічному напрямку.

```
torusNode.look(at: endNode.position, up: directionToViewer, localFront: SCNVector3(1, 0, 0))
```

Далі додаємо вузол тору до кореневого вузла сцени AR, щоб він був видимим у сцені.

```
sceneView.scene.rootNode.addChildNode(torusNode)
```

Після цього встановимо поточний статус текстового поля statusTextView, використовуючи значення структури Measurement для визначення статусу.

Створимо текстову змінну text, в яку запишемо різні виміри та використаємо форматування рядків для відображення чисел з плаваючою комою з заданою точністю.

Всі вимірювані значення будемо виводити в сантиметрах, тому значення довжин будемо множити на 100, а площ на 10000. Записувати в text будемо довжини діаметра, кола і півкола, а також площі кола та півкола. Після чого за

допомогою statusTextView.text = text присвоїмо написаний нами текст з вимірами до властивості text об'єкта statusTextView. Таким чином, вміст statusTextView буде оновлений згідно з новим текстом text.

В результаті, statusTextView буде містити текст з вимірами відстаней та площі для певного об'єкта.

```
var text = "Length of\t\t\t\t·Diameter: \(String(format: "%.2f cm", measurement.distance * 100.0))"
text += "\n·Circle: \(String(format: "%.2f cm", measurement.lengthOfCircle * 100.0))"
text += "\t\t\t\t·Semicircle: \(String(format: "%.2f cm", measurement.lengthOfSemicircle * 100.0))"
text += "\nArea of \n·Circle: \(String(format: "%.2f cm²", measurement.areaOfCircle * 10000.0))"
text += "\t\t·Semicircle: \(String(format: "%.2f cm²", measurement.areaOfSemicircle * 10000.0))"

statusTextView.text = text
```

Тепер, коли if закінчився, ідемо в else, і якщо змінна touchesOnButtonCounter не дорівнює 1, це означає, що користувач або не натиснув на кнопку для вимірювання або натиснув другий раз для закінчення вимірювання.

В такому випадку ми встановлюємо значення змінної isMeasuring false, що означає припинення вимірювання і обнуляємо лічильник нижимань, тобто змінна touchesOnButtonCounter скидається на 0.

```
isMeasuring = false
touchesOnButtonCounter = 0
```

Далі ми виходимо з цієї перевірки і йдемо на else у випадку, якщо поточний стан відстеження не є "READY TO MEASURE".

Тоді ми обнуляємо лічильник кадрів для виведення позначки notReady, тобто змінна seccount скидається на 0.

```
seccount = 0
```

Поле statusTextView та aim встановлюються як приховані, встановлюючи isHidden в true. Це означає, що на екрані не відображатимуться відповідні елементи.

```
statusTextView.isHidden = true
aim.isHidden = true
```

А поле notReady навпаки робимо видимим і встановлюємо свій текст на поточний стан відстеження, отриманий з функції getTrackigDescription().

```
notReady.isHidden = false
notReady.text = getTrackigDescription()
```

На цьому наша основна програма закінчується. Ця функція відповідає за керування вимірюваннями та відображенням елементів на сцені в залежності від стану відстеження ARKit. Вона виконується періодично під час роботи додатка, щоб забезпечити актуалізацію та відображення потрібних елементів користувачеві.

3. ТЕСТУВАННЯ

3.1. Пристрій та камера, на якій здійснювалося тестування

У даний момент я маю iPhone 8 Plus з iOS 16.4.1 (рис. 3.1). Ця модель підтримує ARKit, що дозволяє використовувати доповнену реальність на цьому пристрої. iPhone 8 Plus має кілька плюсів і мінусів для використання доповненої реальності з ARKit.

Рисунок 3.1 - iPhone 8 Plus

Плюси iPhone 8 Plus для доповненої реальності [22]:

- Обчислювальна потужність: iPhone 8 Plus оснащений потужним процесором A11 Bionic, що забезпечує швидкість і продуктивність для обробки вимогливих AR-додатків.
- Графічна продуктивність: Графічний процесор iPhone 8 Plus гарантує високу якість візуального відображення доповненої реальності з багатодеталізованими і реалістичними віртуальними об'єктами.
- Дисплей: iPhone 8 Plus має великий 5,5-дюймовий дисплей з високою роздільною здатністю, що дозволяє отримати більше іммерсивного досвіду доповненої реальності.

- Камера: Камера iPhone 8 Plus прекрасно підходить для доповненої реальності з ARKit завдяки своїм функціям та можливостям.
- Дві задні камери: iPhone 8 Plus має дві задні камери, що дозволяє використовувати режим портрету та забезпечує точну вимірювальну точність для AR-додатків.
- Оптична стабілізація зображення: Оптична стабілізація зображення допомагає зменшити рухи камери та покращує точність відстеження в AR-додатках.
- Режим освітлення портрету: Цей режим дозволяє керувати освітленням об'єктів на зображенні, що може бути корисним у вимірювальних AR-додатках.
- Висока якість знімків: Камера iPhone 8 Plus забезпечує високу якість знімків з деталізованими кольорами, що може покращити візуальний досвід AR-додатків.

Та все ж таки є і мінуси у iPhone 8 Plus, основним з яких є відсутність технології LiDAR. LiDAR (Light Detection and Ranging) - це технологія, що використовує лазерне випромінювання для вимірювання відстаней та створення точних 3D-зображень оточуючого простору [23]. В контексті доповненої реальності, LiDAR може бути використаний для поліпшення точності відстеження об'єктів і вимірювання розмірів у реальному часі.

Основні переваги технології LiDAR для ARKit:

- Точність: LiDAR вимірює відстань до об'єктів з високою точністю, що дозволяє отримати більш точні вимірювання та відстеження в доповненій реальності. Це особливо корисно для вимірювання розмірів об'єктів і створення реалістичних віртуальних об'єктів.
- Швидкість: LiDAR працює дуже швидко, забезпечуючи миттєву зворотну інформацію про оточуюче середовище. Це дозволяє більш

швидко оновлювати віртуальні об'єкти та взаємодіяти з ними у реальному часі.

- Низька залежність від освітлення: LiDAR працює на основі лазерного випромінювання, що не сильно залежить від рівня освітлення. Це означає, що LiDAR може ефективно працювати як в яскравих, так і в темних умовах, забезпечуючи стабільне вимірювання і відстеження.
- Обробка глибини: За допомогою LiDAR можна отримати точні дані про глибину об'єктів у просторі. Це дозволяє більш реалістично відображати віртуальні об'єкти і ще краще інтегрувати їх з реальними оточуючими об'єктами.

Технологія LiDAR вперше була впроваджена в пристроях Apple з випуском iPhone 12 Pro та iPad Pro. Вона використовується в ARKit для покращення точності відстеження об'єктів, вимірювання розмірів і створення більш реалістичних AR-досвідів. Завдяки технології LiDAR розробники AR-додатків можуть створювати більш точні та іммерсивні AR-додатки, які краще взаємодіють з реальним світом.

Отже, мій смартфон не має вбудованого сканера LiDAR, який може покращити точність вимірювання та відстеження в AR-додатках. Та не зважаючи на це, iPhone 8 Plus є добрим пристроєм для використання доповненої реальності з ARKit, особливо для створення AR-додатків, які не вимагають технології LiDAR.

Інші користувачі, які будуть використовувати даний додаток і мають пристрій з LiDAR просто будуть швидше та іноді точніше визначати відстань.

3.2. Тестування стану відстеження камери

При включенні нашого додатку або після його перезапуску буду проходити процес ініціалізації відстеження і камера не буде готова до

вимірювань перші пару секунд. Перевіримо це на практиці. Запустимо додаток. Після його запуску ми бачимо позначку, яка виводить "INITIALIZING" (рис. 3.2).

Рисунок 3.2 – відбувається ініціалізація відстеження після запуску додатку

Після ініціалізації позначка повинна змінитись і виводити текст про готовність вимірювати "READY TO MEASURE". Разом з цим приховані позначка Aim і текстове поле statusTextView повинні стати видимими на екрані. Текст про готовність вимірювати буде доступний 100 кадрів, що близько 1,5 – 2 секундам, після чого повинен зникнути. Результат можна побачити на рис. 3.3.

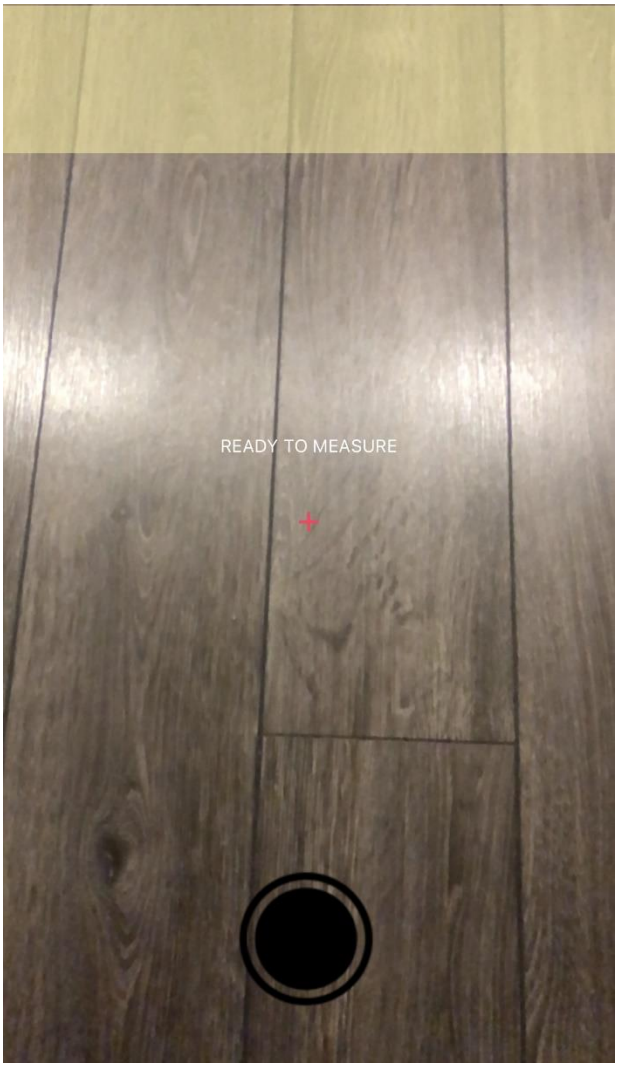

Рисунок 3.3 – додаток готовий вимірювати

Через декілька секунд повідомлення зникає, як повинно бути. Також, позначка повинна виводити "TRACKING LIMITED - Too much camera movement" при надмірному русі камерою. Тому спробуємо побігати ти помахати тримаючи камеру. Результат можна побачити на рис. 3.4.

Рисунок 3.4 – відстеження обмежено через надмірний рух камери

Отже, у випадках руху на автомобілі чи велосипеді, а також під час бігу камера буде обмежувати відстеження і вимірювання.

Також додаток не повинний відстежувати при недостатній кількості деталей знайдених, або при поганому освітленні. Щоб перевірити, чи спрацює запобігання цьому, піднесемо руку до камери. Результат можна побачити на рис. 3.5.

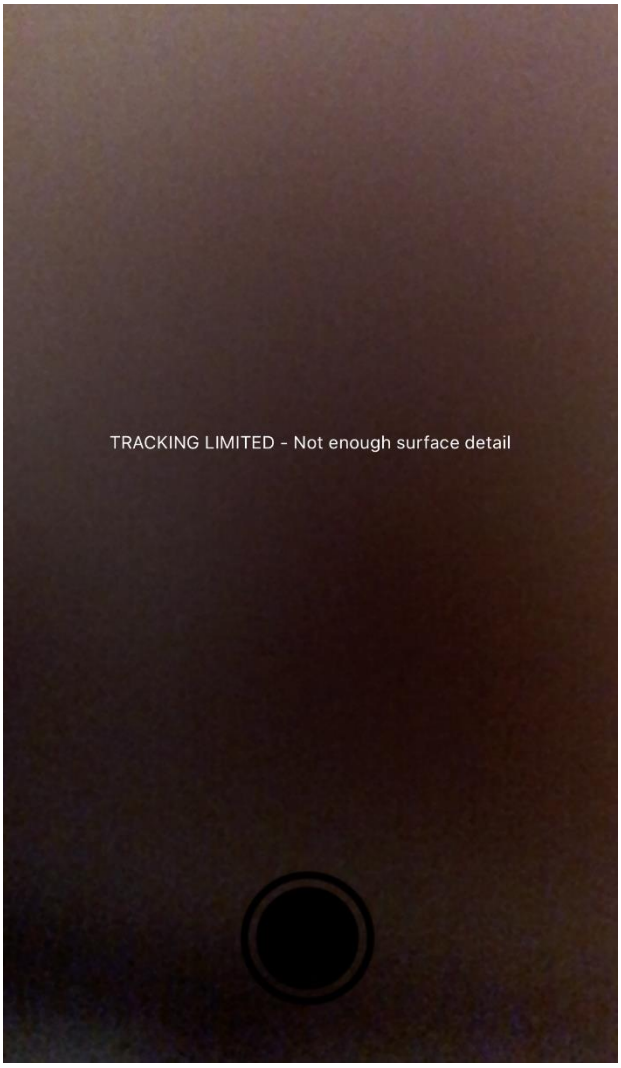

Рисунок 3.5 – відстеження обмежено через недостатню кількість деталей для відстеження

Отже, і в даному випадку додаток спрацював правильно, як він і мав спрацювати.

3.3. Тестування вимірювань

Тепер перейдемо до кінцевого і головного тестування – тестування вимірювання відстаней. Для наглядності спочатку виміряємо довжину лінійки. Результат вимірювання можна побачити на рис. 3.6.

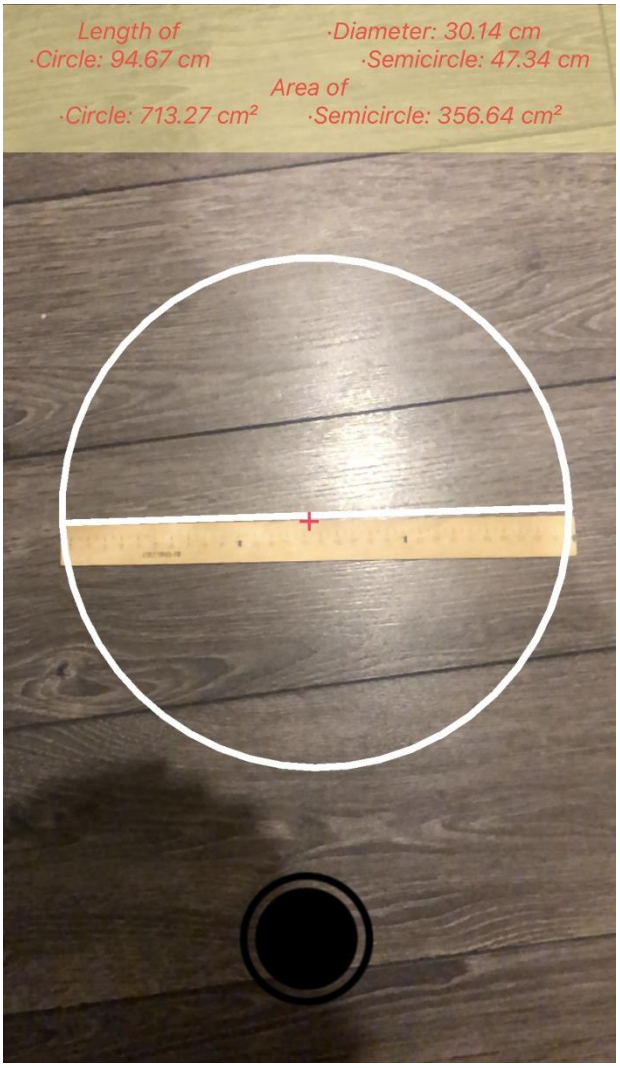

Рисунок 3.6 – вимірювання лінійної відстані

Як ми бачимо, вимірювання є досить точним. Для вимірювання використовувалась лінійка довжиною 30 сантиметрів. Наш додаток визначив, що її довжина 30.14 сантиметрів, що мінімально відрізняється від правдивої. Тепер визначимо довжину та площу кола та півкола. Для цього візьмемо один диск від гантелі. Спочатку визначимо лінійкою його діаметр (рис. 3.7).

Рисунок 3.7 – визначення діаметру диска лінійкою

Ми бачимо, що діаметр диска приблизно дорівнює 15.5 сантиметрів. Тепер, скориставшись калькуляторами, ми знайдемо довжину і площу кола. Отже довжина повинна бути приблизно 48.7 сантиметрів, а площа – 189 сантиметрів квадратних. Тепер перевіримо наш додаток, наскільки точно він визначить ці дані (рис. 3.8).

Рисунок 3.8 – вимірювання довжини і площі диска

З рис. 3.8 бачимо, що дані виміряні лінійкою і обраховані на калькуляторі та виміряні додатком є майже ідентичними. З цього ми можемо зробити висновок, що програма працює на високому рівні та точно визначає лінійні та нелінійні відстані.

ВИСНОВКИ

Отже, за допомогою цього проекту успішно реалізовано створення додатку, який використовує передові технології доповненої реальності для вимірювання відстаней. Цей додаток інтегрується з ARKit, що дозволяє використовувати потужні можливості AR на пристроях Apple, забезпечуючи високу якість та зручність використання.

Одним із ключових аспектів проекту є використання камери вбудованого у пристрої для відстеження об'єктів та точного вимірювання відстаней. Застосування технології LiDAR на пристроях додає ще більшу точність та надійність вимірювань та відстеження об'єктів у AR-додатку.

Під час тестування перевірено, що додаток успішно проходить процес ініціалізації відстеження та стає готовим до вимірювань. При надмірному русі камерою або недостатній кількості деталей для відстеження, відстеження може бути обмежене, що забезпечує правильну роботу додатку в різних ситуаціях.

Результати тестування показали, що додаток точно вимірює як лінійні відстані, наприклад, довжину лінійки, так і нелінійні відстані, такі як діаметр кола та площа. Вимірювання, проведені додатком, майже повністю співпадають з результатами, отриманими за допомогою реальних інструментів, таких як лінійка та калькулятор.

Завдяки цьому проекту користувачі отримують можливість швидко та зручно вимірювати відстані за допомогою своїх пристроїв, що є особливо корисним в різних ситуаціях, як наприклад побутове використання, будівництво, дизайн.

Нозроблений додаток є ефективним та надійним інструментом для вимірювання відстаней з використанням AR-технологій. У цілому, проект є значущий, оскільки він реалізує важливі функціональні можливості, використовує передові технології та забезпечує точність та зручність використання для користувачів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Virtual Reality, VR. URL: https://www.it.ua/knowledge-base/technology-innovation/virtualnaja-realnost-vr (дата звернення: 25.04.2023).
2. Що таке доповнена реальність? VR. URL: https://teach-hub.com/scho-take-dopovnena-realnist/ (дата звернення: 25.04.2023).
3. ЧИМ ВІДРІЗНЯЄТЬСЯ ДОПОВНЕНА (РОЗШИРЕНА) РЕАЛЬНІСТЬ (AR) ВІД ВІРТУАЛЬНОЇ РЕАЛЬНОСТІ (VR)? URL: https://futurenow.com.ua/chym-vidriznyayetsya-rozshyrena-realnist-ar-vid-virtualnoyi-realnosti-vr/ (дата звернення: 25.04.2023).
4. Що таке AR? Поняття доповненої реальності. URL: https://www.adobe.com/ua/products/substance3d/discover/what-is-ar.html (дата звернення: 25.04.2023).
5. Pokémon Go: гра, яка зробила світ ще більш божевільним. URL: https://www.04141.com.ua/news/1305601/pokemon-go-gra-aka-zrobila-svit-se-bils-bozevilnim (дата звернення: 26.04.2023).
6. 12 кращих SDK-пакетів доповненої реальності. URL: https://ulab.sumdu.edu.ua/uk/12-krashhih-sdk-paketiv-dopovnenoi-realnosti (дата звернення: 28.04.2023).
7. В чем разница между Google AR и iPhone 13 AR? URL: https://icoola.ua/ru/blog/googlear-or-iphone13ar/ (дата звернення: 28.04.2023).
8. Все про ARCore. URL: https://forokd.com/uk/arcore/ (дата звернення: 29.04.2023).
9. Vuforia Engine. URL: https://vuforia.mont.com/ (дата звернення: 29.04.2023).
10. Vuforia: Market-Leading Enterprise AR. URL: https://www.ptc.com/en/products/vuforia (дата звернення: 01.05.2023).
11. Використання програми «Мірило» на iPhone, iPad і iPod touch. URL: https://support.apple.com/uk-ua/HT208924 (дата звернення: 05.05.2023).
12. MeasureKit - AR Ruler Tape. URL: https://apps.apple.com/ru/app/measurekit-ar-ruler-tape/id1258270451 (дата звернення: 05.05.2023).
13. AirMeasure - AR Tape & Ruler. URL: https://apps.apple.com/us/app/airmeasure-ar-tape-ruler/id1251282152 (дата звернення: 05.05.2023).

14. Що таке OpenAI XCode? Особливості, переваги та альтернативи. URL: https://hashdork.com/uk/openai-xcode/ (дата звернення: 08.05.2023).
15. Що таке код Visual Studio? - Особливості та переваги - Сфера та кар'єра. URL: https://uk.education-wiki.com/3958588-what-is-visual-studio-code (дата звернення: 08.05.2023).
16. AppCode. URL: https://www.jetbrains.com/ru-ru/objc/features/ (дата звернення: 08.05.2023).
17. JetBrains IntelliJ IDEA. URL: https://itpro.ua/product/jetbrains-intellij-idea/?tab=description (дата звернення: 09.05.2023).
18. SceneKit 3D Programming for iOS: Getting Started. URL: https://www.kodeco.com/23483920-scenekit-3d-programming-for-ios-getting-started (дата звернення: 11.05.2023).
19. Apple Developer Documentation. URL: https://developer.apple.com/documentation (дата звернення: 15.05.2023).
20. Циліндр. Формули та властивості циліндра. URL: https://ua.onlinemschool.com/math/formula/cylinder/ (дата звернення: 25.05.2023).
21. Тороїд - що це таке, визначення та поняття - 2021 - Economy-Wiki.com. URL: https://uk.economy-pedia.com/11036228-toroid (дата звернення: 25.05.2023).
22. IPHONE 8 PLUS: ЙОГО ПЛЮСИ І ... ПЛЮСИ. URL: https://sota.kh.ua/ua/iphone-8-plus-ego-plyusi-i-plyusi-818/obzori-61.html (дата звернення: 29.05.2023).
23. Що таке LiDAR Scanner? URL: https://icoola.ua/blog/sho-take-lidar-scanner/ (дата звернення: 29.05.2023).