МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: «Додаток доповненої реальності для облаштування інтер'єрів меблями»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студентка групи ІТ-61 Оношко Вікторія Валеріївна

Кваліфікаційна робота бакалавра
захищена на засіданні ЕК
з оцінкою _____ «___» _____2020 р.

Науковий керівник _____ к.т.н., Кузнєцов Е. Г.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Голова комісії _____ Шифрін Д. М.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Засвідчую, що у цій дипломній роботі немає
запозичень з праць інших авторів
без відповідних посилань.

Студент _____
(підпис)

Суми-2020

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук
Секція інформаційних технологій проектування
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав. секцією ІТП

_____ В. В. Шендрик
«__» _____ 2020 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Оношко Вікторія Валеріївна

1 Тема роботи *Додаток доповненої реальності для облаштування інтер'єрів меблями*

керівник роботи *Кузнєцов Едуард Геннадійович, к.т.н.,* ,

затверджені наказом по університету від « 14 » травня 2020 р. № 0576-III

2 Строк подання студентом роботи «1» червня 2020 р.

3 Вхідні дані до роботи технічне завдання на розробку додатку доповненої реальності для облаштування інтер'єрів меблями

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз предметної області, моделювання та проектування, реалізація додатку доповненої реальності для облаштування інтер'єру меблями

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) актуальність проблеми, мета та задачі проекту, аналогічні додатки AR, функціональні вимоги, інструменти та засоби реалізації, моделювання програмного продукту, етапи розробки додатку доповненої реальності для облаштування інтер'єрів меблями, висновки

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7.Дата видачі завдання_____01.10.2019_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Ознайомлення та аналіз предметної області	18.11.19 – 22.11.19	
2	Дослідження аналогів	25.11.19 – 27.11.19	
3	Визначення мети проекту	28.11.19 – 29.11.19	
4	Вивчення документації Autodesk ReCap	2.12.19 – 5.12.19	
5	Вивчення документації Unity 3D	6.12.19 – 11.12.19	
6	Планування WBS	12.12.19 – 13.12.19	
7	Планування OBS	16.12.19 – 17.12.19	
8	Створення календарного плану	18.12.19 – 20.12.19	
9	Управління ризиками	23.12.19 – 25.12.19	
10	Розробка структури проекту	02.01.20 – 17.01.20	
11	Реалізація структури проекту в Unity	20.01.20 – 11.05.20	
12	Розробка 3D моделей меблів	24.02.20 – 04.05.20	
13	Розробка функціоналу доповненої реальності	06.04.20 – 24.04.20	
14	Тестування продукту	12.05.20 – 20.05.20	
15	Проектна документація	02.01.20 – 20.05.20	
16	Введення в експлуатацію	21.05.20 – 21.05.20	
17	Архівація проекту	22.05.20 – 22.05.20	

Студент _____ Оношко В.В.
(підпис)

Керівник роботи _____ к.т.н., Кузнєцов Е.Г.
(підпис)

РЕФЕРАТ

Тема роботи «Додаток доповненої реальності для облаштування інтер’єрів меблями»

Пояснювальна записка складається зі вступу, 3 розділів, висновків, списку використаних джерел із 20 найменувань, 4 додатків. Загальний обсяг роботи − 87 сторінок, у тому числі 50 сторінок основного тексту, 2 сторінки списку використаних джерел, 34 сторінок додатків.

Кваліфікаційну роботу бакалавра присвячено розробці додатку доповненої реальності для облаштування інтер’єрів меблями.

У роботі виконано дослідження актуальності поставленої проблеми, а також предметної області, наведена її характеристика, існуючі проблеми та алгоритми.

В роботі проведено аналіз існуючих аналогів рішень у сфері розробки додатків доповненої реальності.

Виконано проектування продукту проекту за допомогою UML на основі проведеного структурно-функціонального аналізу продукту проекту.

Крім того, описано практичну частину реалізації проекту, детально зображено процес створення 3D моделей меблів та процес розробки додатку доповненої реальності.

Результатом проведеної роботи є розроблений додаток доповненої реальності для облаштування інтер’єрів меблями.

Практичне значення роботи полягає у використанні додатку для проектування інтер’єрів.

Ключові слова: C#, .NET, Unity3D, Augmented Reality, Photogrammetry, 3D-модель.

ЗМІСТ

ВСТУП

Останнім часом все більше доповнена реальність входить у наше повсякденне життя для того, аби зробити його зручнішим та цікавішим. У додаток камери будь-якого сучасного смартфону включені AREmoji, при трансляції футбольних матчів зображується позиція виконання штрафного удару, створюються безліч розважальних ресурсів та ігор, які дозволяють познайомитися з можливостями поєднання реальності з будь-якими віртуальними елементами.

Поширення доповненої реальності разом із покращенням обчислювальної потужності та датчиків апаратної платформи смартфонів та планшетів дозволяють накладати будь-які цифрові дані на одержувану картинку в реальному часі у будь-якій сфері. Зважаючи на це, можемо вважати розробку додатку доповненої реальності актуальною.

Дані для відображення можуть бути представлені як у 2D, так і 3D форматі, в залежності від потреб користувачів. На сьогоднішній день, коли технології 3D моделювання достатньо розвинені, нам хочеться отримувати все більш реалістичні об’єкти. Цього можна досягти створюючи детальні моделі вручну за допомогою спеціального ПЗ чи онлайн редакторів, або ж автоматично, за допомогою 3D сканерів. Виготовлення моделей вручну є достатньо довгим та кропітким процесом, особливо якщо ми хочемо отримати реалістичний результат, а сканування зазвичай вимагає спеціального обладнання.

Ще однією цікавою технологією, яка однозначно привертає увагу є фотограмметрія. Вона дозволяє отримувати 3D-скан будь-якого об’єкту, використовуючи лише фотознімки. Таким чином, маючи під рукою смартфон, ми отримуємо можливість створення 3D моделі. Фотограмметрія має так само безліч застосувань, дозволяє отримати тривимірні моделі, масиви точок як невеликих об’єктів, так і цілих архітектурних споруд.

Зіштовхнувшись з проблемою вибору нових меблів, стало очевидним, що сучасній людині не завадила б можливість побачити заздалегідь як виглядатиме та чи

інша річ в інтер’єрі перед її придбанням. Ідея проекту полягає в тому, щоб перетворити за допомогою фотограмметрії набори фотографій реальних меблів на їх 3D моделі та за допомогою інструментів доповненої реальності дозволити розташовувати отримані моделі в інтер’єрі.

Об'єктом дослідження дипломного проектування є послуга створення інтер'єру приміщень шляхом підбору і розміщення меблів.

Предметом дослідження є використання технології доповненої реальності для створення віртуального прототипу інтер'єру.

На основі ідеї було сформовано мету даної роботи, яка полягає у розробці додатку доповненої реальності для облаштування інтер’єрів меблями. Основними задачами для досягнення даної мети з якими ми зустрінемося в ході виконання розробки продукту проекту є:

- детальний аналіз предметної області та аналогічних продуктів;
- вибір технологій розробки продукту;
- розробка набору моделей меблів;
- розробка внутрішнього модулю додатку;
- розробка інтерфейсу додатку;
- проведення тестування продукту.

Практичне значення даної роботи полягає у застосуванні для планування інтер’єру, вона допоможе користувачам уникати покупки меблів, які в результаті можуть виявитися недостатньо підходящими для створеного інтер’єру.

Дана розробка може бути корисна як звичайним меблевим магазинам, так і інтернет магазинам, може підвищити продажі, через те що клієнти не будуть боятися замовляти продукцію, яка може не сподобатися в реальності, зменшити кількість повернень товару, тому що клієнти зможуть заздалегідь впевнитись у виборі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальна характеристика предметної області

Доповнена реальність є середовищем із прямими або непрямими доповненнями до фізичного світу віртуальних даних. Об'єкти доповненої реальності додаються в режимі реального часу за допомогою цифрових пристроїв – планшетів, смартфонів, смарт-окулярів або аксесуарів зі спеціальним програмним забезпеченням в якості провідника [1].

Доповнена реальність на сьогоднішній день використовується в різних сферах діяльності: освіті, медицині, туризмі, авіації, військовій справі, маркетингу, дизайні та іграх. Доповнена реальність допомагає дітям дізнаватися більше про світ, за допомогою камери є можливість побачити детальну інформацію про різні предмети. Також допомагає у підготовці військових, шляхом навчання на полігонах із використанням доповненої реальності та у підготовці нових пілотів, може виконувати функції штурмана. У галузі маркетингу доповнена реальність допомагає продавати широкий спектр товарів і послуг, демонструючи ці товари за допомогою спеціальної реклами. Туристу доповнена реальність може допомогти знайти шлях або ж знайти та переглянути інформацію про архітектурну споруду чи інші туристичні об’єкти, на які користувач наводить свій мобільний телефон. Дизайнерам, завдяки доповненій реальності, можна комфортно презентувати свої роботи, навіть архітектуру складної будівлі будь-яких розмірів.

Додатки доповненої реальності допомагають людині зосередитись на елементах зображення камери; покращити сприйняття оточуючого світу, надаючи необхідну інформацію, що додається до існуючого зображення у вигляді тексту або візуальної моделі чи зображення [2].

1.2 Огляд останніх досліджень і публікацій

Оскільки технологія доповненої реальності є відносно молодою, існує не дуже велика кількість літературних джерел, присвячених вирішенню питань, близьких до поставленої мети проектування. Проте, ми маємо в розпорядженні безліч відео-уроків з розробки елементарних застосувань доповненої реальності. Крім того, існує велика кількість статей та інформаційних блогів, які в достатній мірі розкривають дану тему.

З метою покращення існуючих інструментів для розробки додатків з застосуванням доповненої реальності проводяться дослідження, піднімаються питання на різноманітних конференціях. Теми, близькі до поставленої мети є обговорюваними, тож є очікуваним зростання кількості літературних джерел.

Існує декілька технологій, що можуть бути використані для роботи AR. Однією з яких є доповнена реальність на основі маркерів, яка також називається розпізнаванням зображень. Даний тип технології потребує камеру та візуальний маркер, такий як QR код, який відображає записаний всередині нього контент лише в результаті зчитування сенсором. Це дає змогу виділити із реального світу віртуальні елементи.

Іншою технологією є безмаркерна доповнена реальність, також відома як координатно-, або GPS-орієнтована. Для надання даних про ваше місцезнаходження, може використовуватись Глобальна система позиціонування (GPS), датчик швидкості або цифровий компас, якими обладнано пристрій. Через масове поширення різноманітних гаджетів дана технологія в даний час використовується найчастіше. Найбільш поширені випадки використання – це відображення напрямків, знаходження потрібних локацій, наприклад кафе чи офісу, або ж у додатках на основі місцезнаходження.

Ще одна технологія – доповнена реальність на основі VIO (Visual Inertial Odometry). Візуальна інерціальна одометрія є технологією, яка дозволяє відстежувати положення, і разом із цим, знаходити орієнтири в просторі за допомогою датчиків та камери. Через це є можливість створити 3D-модель простору, оновити її в реальному

часі, визначити положення в ній, передати ці дані у всі додатки та накласти додаткові шари поверх неї. Дана технологія має дійсно унікальні можливості: вимірювати відстань, вставляти різні предмети в простір та взаємодіяти з ними. Технологія VIO може виявитись найперспективнішою технологією в AR, нині нею користуються такі гіганти, як Google в Project Tango та Apple в ARKit [1].

1.3 Аналіз програмних продуктів - аналогів

За запитом пошуку схожих рішень було отримано декілька аналогічних додатків. Перед тим як розпочати роботу над проектом було проведено аналіз існуючих аналогів продукту.

Ikea Place – цей додаток отримав велику кількість реклами при запуску, адже це був перший подібний додаток, випущено його було у 2017 році для платформи iOS [5]. Ikea Place дозволяє віртуальне розміщення 3D моделей у власному просторі. Для відображення просто потрібно погуляти кілька десятків секунд по приміщенню, для того аби камера сприйняла підлогу, а потім отримати доступ до віртуальних предметів меблів – таких як, диван або ж ліжко, стілець, письмовий стіл або шафа для одягу і розташувати його там, де необхідно. Меблі будуть відображені на тому місці, де вони були встановлені, в той час як можна вільно переміщуватися по приміщенню і розглядати їх з різних точок [3].

Крім того, даний додаток дозволяє виконувати візуальний пошук, який при наведенні камери на предмет меблів визначає яким саме продуктом Ikea він є, або найбільше схожий.

Версія для платформи Android розроблена з допомогою Google ARCore, що є аналогом IOS ARKit. За функціоналом версія для android майже повністю ідентична версіїї IOS і надає можливість обрати меблі з каталогу виробника. Проте, не всі користувачі даної платформи можуть використовувати додаток, він має список сумісних пристроїв, таким чином навіть із останніми версіями android можна не мати

можливості скористатися додатком [4-5]. Розміщений на GooglePlay додаток Ikea Place наведено на рис. 1.1, також можна переконатися у тому що додаток несумісний із пристроями. Інтерфейс дадатку, такий як його показано у GooglePlay, представлено на рис. 1.2.

Рисунок 1.1 – Ikea Place на GooglePlay

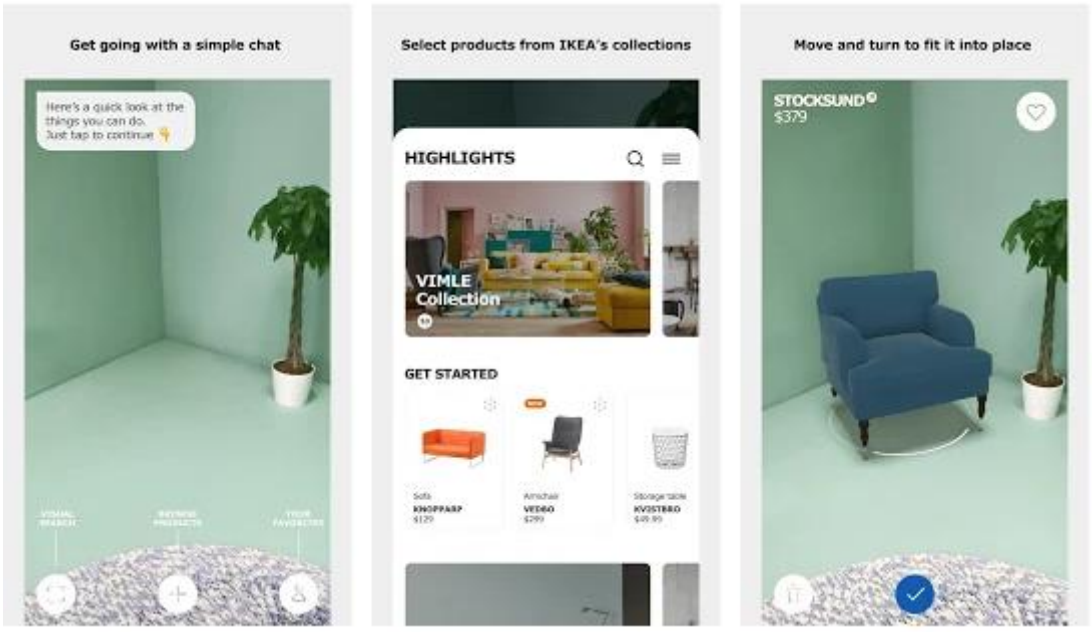

Рисунок 1.2 – Ikea Place на GooglePlay

Housecraft – додаток схожий на попередній, доступний він лише для iPhone. Дозволяє зберегти конфігурації кімнати та спробувати розмістити іх в іншому місці, що є чудовим рішенням для пошуку житла, реорганізації приміщень та іншого. Має великий асортимент кімнатних рослин, що дозволяє побачити яким чином може

виглядати кімната після «озеленіння». Із додаванням меблів та рослин можна отримати уявлення наповненого меблями приміщення. Проте, Housecraft позиціонує себе як розважальний додаток, він не містить реальних обьектів, дає можливість як завгодно змінювати розміри моделей, крім цього можна зруйнувати свій результат, запустивши торандо, яке змусить всі обьекти злетіти вгору. Також містить вбудовану можливість відеозапису та зйомки, що дозволяє легко поділитися результатом [6]. Запропоновані можливості даного додатку представлено на рис. 1.3.

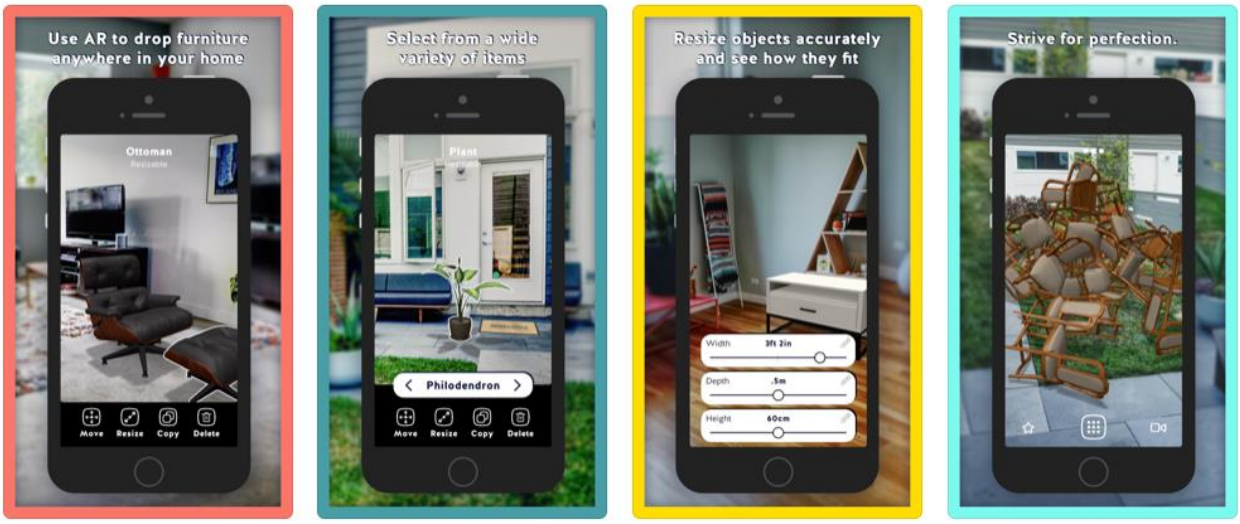

Рисунок 1.3 – Можливості додатку Housecraft

Myty – ще один додаток доповненої реальності зі сфери дизайну інтер'єрів, даний додаток доступний як на Android, так і на iOS. Він дозволяє користувачам смартфонів переробляти свій будинок. Це здебільшого корисний ресурс для всіх любителів дизайну інтер'єру. Все, що необхідно для натхнення створення інтер'єру приміщення, доступне в даному додатку: можливість переглянути тисячі фотографій вже створених інтер'єрів, знайти красиві предмети меблів від найвідоміших світових брендів та отримати доступ до портфоліо кращих дизайнерів меблів. Можна слідкувати за улюбленими брендами та дизайнерами та зберігати ідеї, щоб у будь-який момент отримати доступ до них. Додаток виступає в ролі портфоліо для дизайнерів інтер'єрів.

Також є можливість створення віртуальної кімнати, з різними текстурами стін та підлоги, з додаванням дверей та вікон [7]. Можливості додатку представлено на рис. 1.4.

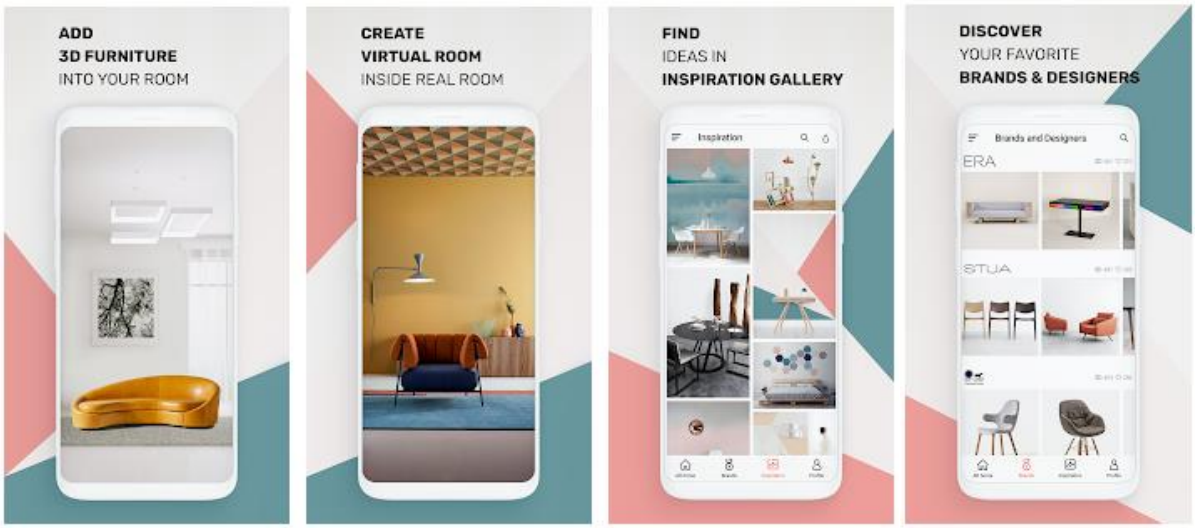

Рисунок 1.4 – Можливості додатку Myty

В результаті аналізу програмних продуктів було створено порівняльну характеристику аналогів, яку представлено в табл. 1.1.

Таблиця 1.1 – Порівняльна характеристика аналогів

Критерій	Ikea Place	Housecraft	Myty
IOS	+	+	+
Android	+-	-	+
Безкоштовне використання	+	+	+
Реклама	-	-	-
Реєстрація	Профіль IKEA	Не потребує	Необхідна для деяких функцій
Оновлюваність	10 грудня 2019 р.	1 листопада 2017 р.	11 квітня 2020 р
Популярність (встановлення)	100 000+	30 000+	5 000+
Оцінка PlayMarket /App Store	3.2/4.7	4.5(App Store)	4.2/4.2

На відміну від вже існуючих аналогів очікуваним результатом є додаток для платформи Android, особливістю якого можна вважати те, що відображувані моделі меблів створюються шляхом тривимірного сканування екземплярів меблів за допомогою фотограметрії, а тому є реальним відображенням предметів у просторі.

1.4 Постановка задачі

1.4.1 Мета та задачі

Метою проекту є розробка додатку доповненої реальності для облаштування інтер'єрів меблями, за допомогою якого користувач зможе легко обрати бажані предмети інтер'єру із наданого переліку, переглянути їх на екрані та розташувати в будь-якій частині свого інтер'єру за допомогою технологій доповненої реальності. Надані користувачам моделі мають відображати реальні об'єкти.

Після запуску програмного продукту користувач матиме можливість обрати бажаний предмет інтер'єру зі списку-каталогу, перейшовши до режиму доповненої реальності розташувати його в будь-якій частині кімнати направляючи камеру в потрібний простір приміщення. Після цього користувач також може додавати інші предмети до тих, що вже знаходяться на сцені.

Основними задачами для досягнення даної мети з якими ми зустрінемося в ході виконання розробки продукту проекту є:

- детальний аналіз предметної області та аналогічних продуктів, вибір технологій розробки;
- розробка набору моделей меблів;
- розробка інтерфейсу та внутрішнього модулю додатку;
- проведення тестування продукту.

Для більш детального опису технічне завдання на розробку продукту проекту наводиться у додатку А.

Технологія доповненої реальності швидко розвивається в наші дні і є найбільш перспективною технологією, за словами генерального директора компанії Apple [2].

Одним з найбільш важливих моментів в розробці системи доповненої реальності є створення відповідного інтуїтивно зрозумілого та зручного інтерфейсу між користувачем і віртуальними об'єктами.

Після огляду найбільш відомих ігрових рушіїв для розробки продукту проекту було обрано Unity3D. Ця платформа допомагає створити власний продукт використовуючи функції 2D та 3D розробки та працюючи разом зі командою. Unity дозволяє імпортувати ресурси з багатьох 3D додатків, таких як Maya або Blender, і пропонує широкий спектр ресурсів, які можна отримати зі Asset Store. Він є унікальним рушієм, який містить в собі інструменти, що дозволяють створювати AR-додатки під різні платформи. Технологія Unity3D використовується для розробки більшості відомих AR проектів.

Серед наборів інструментів для розробки AR додатків з Unity найбільш відомими є: ARKit, ARCore, Vuforia AR Toolkit, EasyAR. Провівши порівняння технологій за певними критеріями, була сформована табл. 1.2.

Таблиця 1.2 – Порівняльна таблиця AR SDK

	ARKit	ARCore	Vuforia AR	EasyAR
Підтримувані платформи	iOS 11	Android 7.0 і вище, iOS 11 або вище	Android, iOS, UWP і Unity Editor.	Android, iOS, UWP, Windows, Mac і Unity Editor.
Вартість	безкоштовно / $99 щорічна програма для розробників	безкоштовно	безкоштовно з водяним знаком Vuforia	безкоштовно
Максимальне захоплення відстані (м)	1.5 / 5	1.0 / 3	1.2 / 3.7	0.9 / 2.7
Мінімальний кут розпізнавання	30	50	30	35

Продовження таблиці 1.2 – Порівняльна таблиця AR SDK

Стабільність розпізнавання нерухомого маркера	9	9	10	7
Стабільність розпізнавання рухомого маркера	7	6	6	3

Після проведеного аналізу було вирішено, що для розробки продукту проекту буде використано Vuforia AR, оскільки дана технологія дозволить створити універсальний додаток, а також є безкоштовною та легкою альтернативою, яка може використовуватися поверх ARCore, ARKit та Microsoft Hololens, надаючи додаткові функції для вмісту AR.

Розширення Vuforia Unity забезпечує:

– Vuforia Prefabs та сценарії компонентів для створення додатків Vuforia в Unity;

– Бібліотеки API Vuforia, які відкривають API Vuforia в C#;

– Підтримка всіх функцій і відстежуваних типів Vuforia;

– Високорівневий доступ до апаратних засобів пристрою, таких як камера пристрою.

Vuforia також є провідною технологією в галузі розробки додатків доповненої реальності для промисловості завдяки використаної в ній кращої технології машинного зору, потужних функцій відстеження і широти підтримки платформи. Враховуючи більш ніж 600 000 зареєстрованих розробників і сотень найбільших корпоративних клієнтів по всьому світу в більш ніж 30 галузях – жодна інша технологія не використовується ширше для застосування потужних можливостей доповненої реальності [9].

Для створення моделей було обрано можливості Autodesk Recap. Це рушій фотограмметрії, який може обробити до 1000 фотографій. З його допомогою користувач може завантажити на сервер Autodesk комплект фотографій та отримати

в результаті триангульовану модель об'єкта з текстурами високої якості. Програмне забезпечення ReCap просте у використанні, і оскільки обробка інформації здійснюється на серверах компанії Autodesk, це дозволяє використовувати сервіс навіть на смартфонах. Один з основних сервісів обробки даних – це Auto Cleanup, який автоматично визначає та видаляє небажані точки, такі як люди або рухомі об’єкти, які були сфотографовані випадково. Структура проекту дозволяє розбити сканування на фрагменти та працювати лише з потрібними даними в будь-який момент часу. А студентська ліцензія дозволяє використовувати хмарні технології даного сервісу.

1.4.2 Вибір засобів реалізації

Мова C # дуже популярна серед розробників. Особливості мови C # дуже важливі для програмістів. Вона містить більшість функцій, яких потребує програміст сьогодні. Розробники ігрових додатків дуже часто використовують цю мову для розробки ігор за допомогою рушія Unity, який є найпопулярнішим сьогодні.

Для реалізації додатку доповненої реальності було вирішено обрати мову програмування C# для написання скриптів Unity.

Вбудовані компоненти Unity дуже універсальні, проте оскільки завжди хочеться за рамки того, що вони можуть забезпечити, необхідно реалізувати власну логіку. Скрипти використовуються для реалізації власної логіки та поведінки, які потім додаються як компоненти до GameObjects. Кожен сценарій здійснює свій зв'язок із внутрішніми функціями Unity, реалізуючи клас, який походить від вбудованого класу під назвою MonoBehaviour.

Сценарій є важливим елементом у всіх розробках, які створюються в Unity. Більшість програм потребують сценаріїв, щоб реагувати на вхідні дані гравця та організувати події в геймплеї щоб вони відбувалися тоді, коли вони повинні відбутися. Крім цього, є можливість застосовувати скрипти для створення графічних ефектів, контролю фізичної поведінки об'єктів або навіть для реалізації спеціальної системи AI для персонажів у грі.

Мова C# найкраще підходить для створення сценаріїв, з багатьох причин. Однією з них є те, що Unity використовує Mono, який є крос-платформою .NET Framework. Мова C# є основною мовою .NET, і всі бібліотеки Unity створені за допомогою коду C#. Сказати, що C# є мовою Unity, не було б перебільшенням. Unity останнім часом дають зрозуміти, що вони вважають C# єдиною мовою для використання з рушієм, який стрімко розвивається. З впровадженням C# job system та ECS Unity просуває все, що можна робити з C# все далі та далі, а новий компілятор Burst робить це швидше, ніж будь-коли раніше [13].

Крім того, існує вичерпна документація Unity із написання сценаріїв на мові C#. Посилання на сценарії організовуються відповідно до класів, доступних для сценаріїв, які описані разом з їх методами, властивостями та будь-якою іншою інформацією, що стосується їх використання. Сторінки документації також широко забезпечені прикладами коду.

2 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ

2.1 Проектування інформаційної системи

Після проведення аналізу предметної області, програмних продуктів - аналогів, визначення мети та задач проекту, а також вибору засобів реалізації, необхідним етапом є проектування інформаційної системи. Представлення функціоналу майбутньої розробки доцільно представити у вигляді Use case та IDEF.

2.1.1 Моделювання використання інформаційної системи

Модель використання включає в себе опис акторів, опис варіантів використання та діаграми сценаріїв використання в UML (Use case diagram). Суть даного типу діаграм полягає в тому, що система, яка проектується, представлена як група акторів, які за допомогою варіантів використання взаємодіють із системою. Діаграма варіантів використання відображає функціональне призначення проектованої програмної системи.

Акторами в даному випадку визначено:

– Customer – замовник, або користувач, який використовує систему.

– AR SDK – фреймворк, завдяки якому виконується реалізація режиму доповненої реальності додатку.

– Cloud Storage – сховище даних, де зберігаються моделі.

– Designer – розробник, який створює систему

– Autodesk ReCap – програмне забезпечення 3D сканування та захоплення реальності для створення моделей.

Після визначення акторів системи, необхідно сформувати перелік усіх варіантів використання, з якими будуть взаємодіяти визначені актори. Варіанти використання

позначають виконання певних операцій або дій та відповідають вимогам до системи, які визначені в Додатку А.

Варіантами використання для даної системи є:

– Створення 3D моделей меблів
– Перегляд списку меблів
– Вибір 3D моделі зі списку
– Перехід до перегляду моделі в режимі AR
– Редагування положення меблів

За результатами сформованих варіантів використання та акторів системи було розроблено діаграму варіантів використання, яку наведено на рис. 2.1.

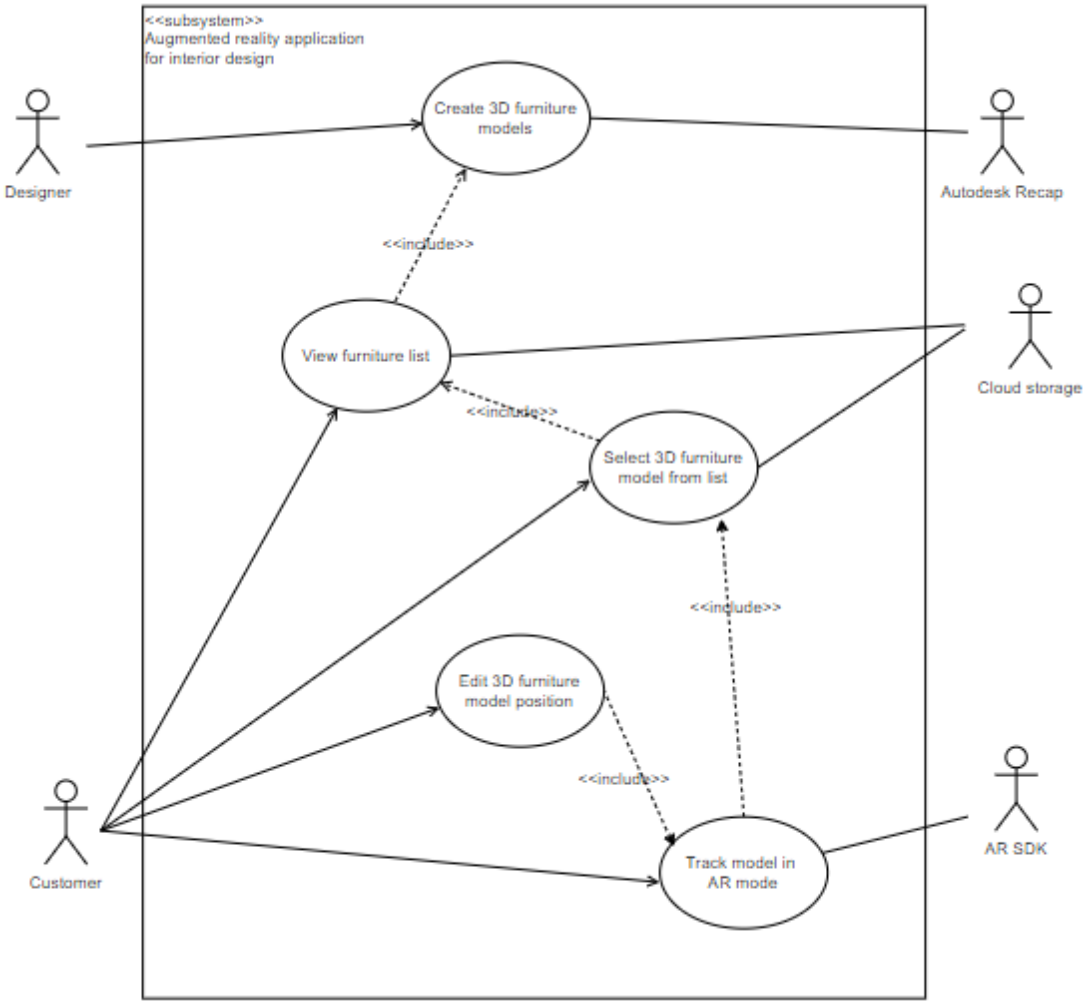

Рисунок 2.1 – Діаграма варіантів використання

2.1.2 Моделювання інформаційної системи в IDEF0

IDEF0 – методологія описання функціонального аспекта модельованої системи, основу якої складає графічна мова описання процесів, призначена для формалізації і опису бізнес-процесів.

Основними компонентами даних, які представлені на даній діаграмі є:

– Вхідні: ввідні дані, що визначають певні задачі на стадії ініціалізації продукту.

– Вихідні: дані, що виводять результат, отриманий в ході реалізації.

– Управління: механізми управління (положення, інструкції тощо).

– Механізми: усе що використовується для того, щоб виконати необхідну роботу (програмне, апаратне забезпечення, команда проекту) .

Контекстна діаграма – це діаграма найвищого рівня, що загально представляє систему, і пов’язує її іншими елементами інтерфейсу за допомогою стрілок.

В результаті проведення аналізу було сформовано перелік даних для контекстної діаграми:

– Вхідні: технічне завдання на розробку проекту, вимоги користувача.

– Вихідні: розроблений додаток та набір проектної документації, як звіт із виконаної роботи.

– Управління: набір фотографій предметів меблів, методологія створення Unity додатків, методологія створення проектів Autodesk ReCap.

– Механізми: команда проекту, та все необхідне технічне та апаратне забезпечення.

Контекстну діаграму процесу розробки додатку доповненої реальності для облаштування інтер’єрів меблями наведено на рис. 2.2.

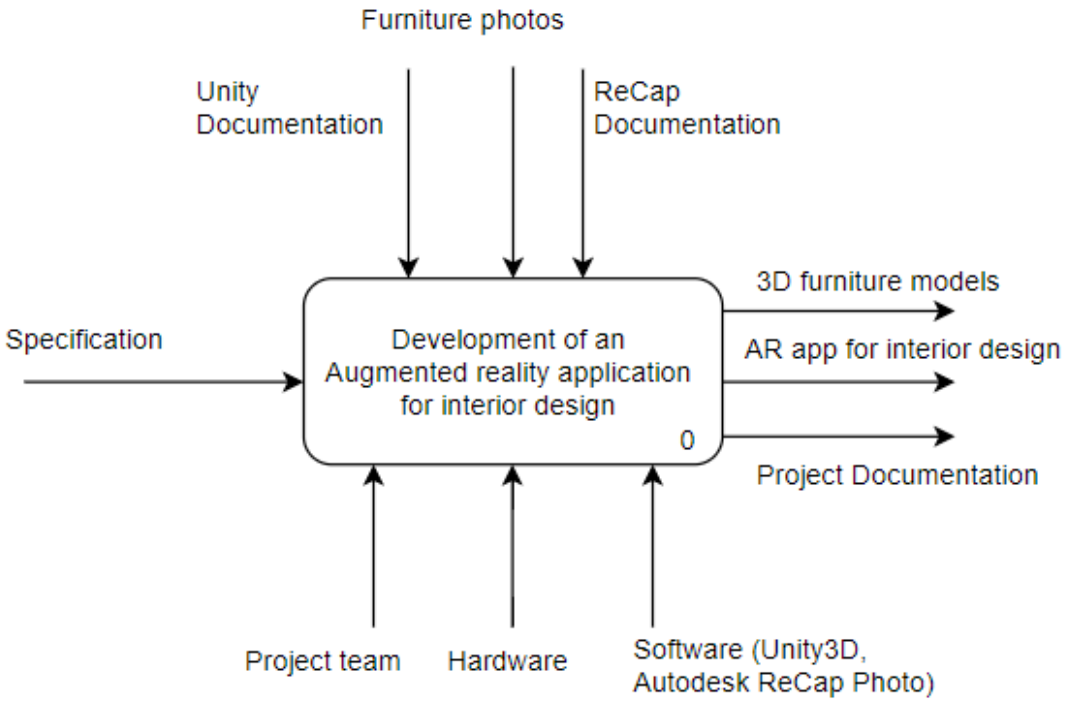

Рисунок 2.2 – Контекстна діаграма

Далі проводиться процес декомпозиції, який надає можливість ознайомитись із послідовністю виконання робіт проекту. Під час декомпозиції було виділено 3 основних етапи, які складають діаграму другого рівня, а саме:

– Розробка тривимірних моделей меблів

– Розробка інтерфейсу додатку

– Розробка основного модулю додатку

Під час опису першого етапу було сформовано наступний перелік даних:

– Вхідні: технічне завдання на розробку проекту, вимоги користувача.

– Вихідні: тривимірні моделі меблів.

– Управління: набір фотографій предметів меблів, методологія створення проектів Autodesk ReCap.

– Механізми: команда проекту, технічне забезпечення, Autodesk ReCap Photo.

Далі так само було сформовано перелік даних другого етапу:

– Вхідні: технічне завдання на розробку проекту, вимоги користувача.

– Вихідні: розроблений та реалізований інтерфейс додатку.

– Управління: методологія створення Unity додатків.

– Механізми: команда проекту, технічне забезпечення, Unity3D.

Третій етап є завершальним і представляє процес розробки внутрішнього модулю додатку. При формуванні третього етапу було виділено наступний перелік даних:

– Вхідні: технічне завдання на розробку проекту, вимоги користувача.

– Вихідні: розроблений додаток та набір проектної документації, як звіт із виконаної роботи.

– Управління: методологія створення Unity додатків.

– Механізми: команда проекту, технічне забезпечення, Unity3D.

Діаграму другого рівня наведено на рис. 2.3.

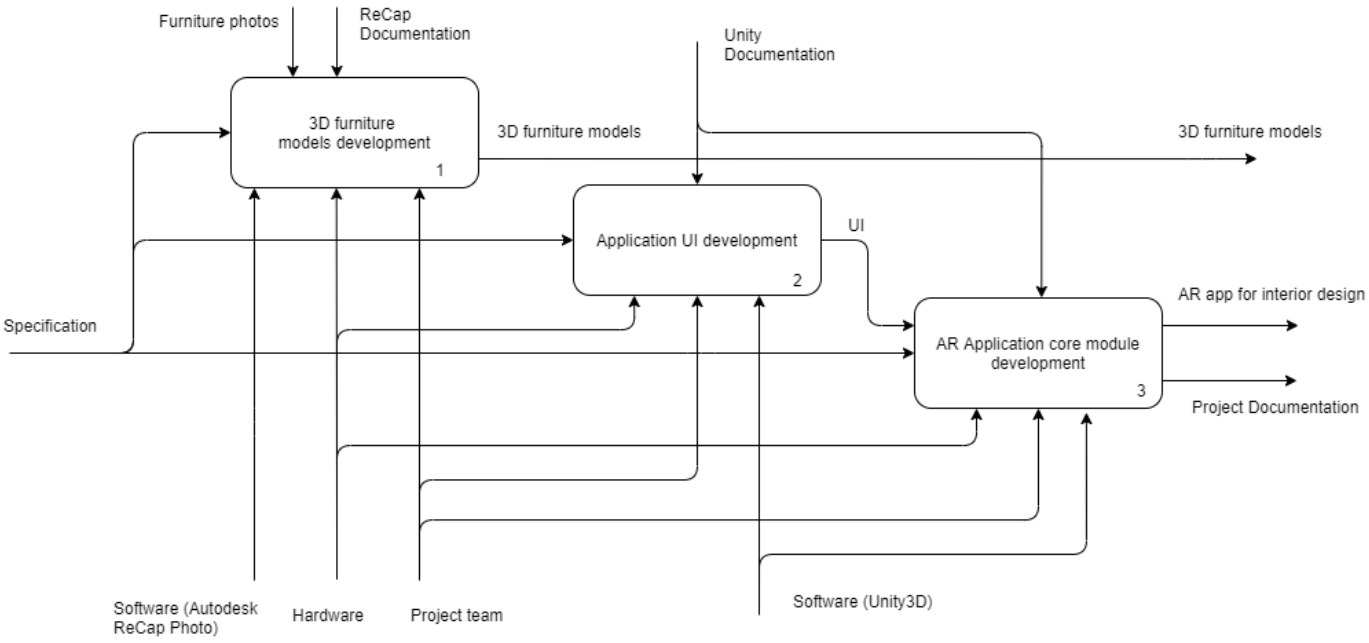

Рисунок 2.3 – Декомпозиція діаграми A-0

Наведені контекстна діаграма та діаграма декомпозиції представлені з точки зору розробки додатку, з точки зору функціонування дані діаграми представлено в Додатку Б на рис. Б.11 – Б.12.

Також проводилось планування робіт проекту, під час якого було проведене формування ієрархічної структури робіт (WBS) та організаційної структури проекту (OBS), що визначають роботу кожного з учасників проекту на кожному етапі ієрархії плану робіт. Розроблювався календарний план проекту у вигляді діаграми Ганта та PDM мережі. Визначалися ризики та формувався план управління ризиками. Повний опис планування робіт проекту наведено у Додатку Б.

3 РЕАЛІЗАЦІЯ

Для реалізації мети продукту проекту необхідно провести реалізацію 3D моделей меблів, реалізацію інтерфейсу та основного модулю додатку доповненої реальності. Створюваний додаток отримав назву InteriAR.

3.1 Реалізація 3D моделей додатку

Створення набору 3D моделей меблів відбувається із використанням рушія фотограмметрії Autodesk ReCap. Для отримання тривимірного фото-скану створюється необхідна кількість фотографій об’єкта.

Для правильного відтворення моделі, фотографії необхідно створювати, пересуваючись навколо об’єкта по колу, при цьому не змінюючи його позицію та уникаючи затінення. Найкраще, фотографувати з інтервалом 5° навколо предмета, також згори та знизу з таким самим інтервалом. Фотографії повинні мати 40 – 70% перекриття.

В результаті проведення фотозйомки предмета, отримано галерею фото, наприклад, як показано на рис. 3.1

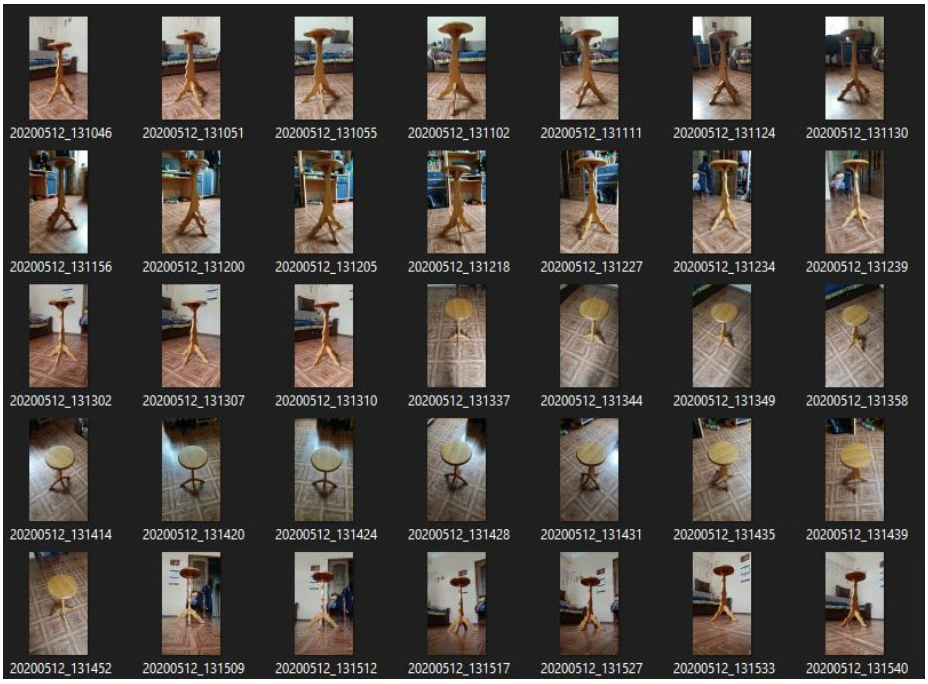

Рисунок 3.1 – Галерея фотографій

Наступним кроком є обробка отриманих фотографій сервером Autodesk. Для цього необхідно створити новий проект у додатку Autodesk ReCap, та додати усі фото до проекту (Рис. 3.2).

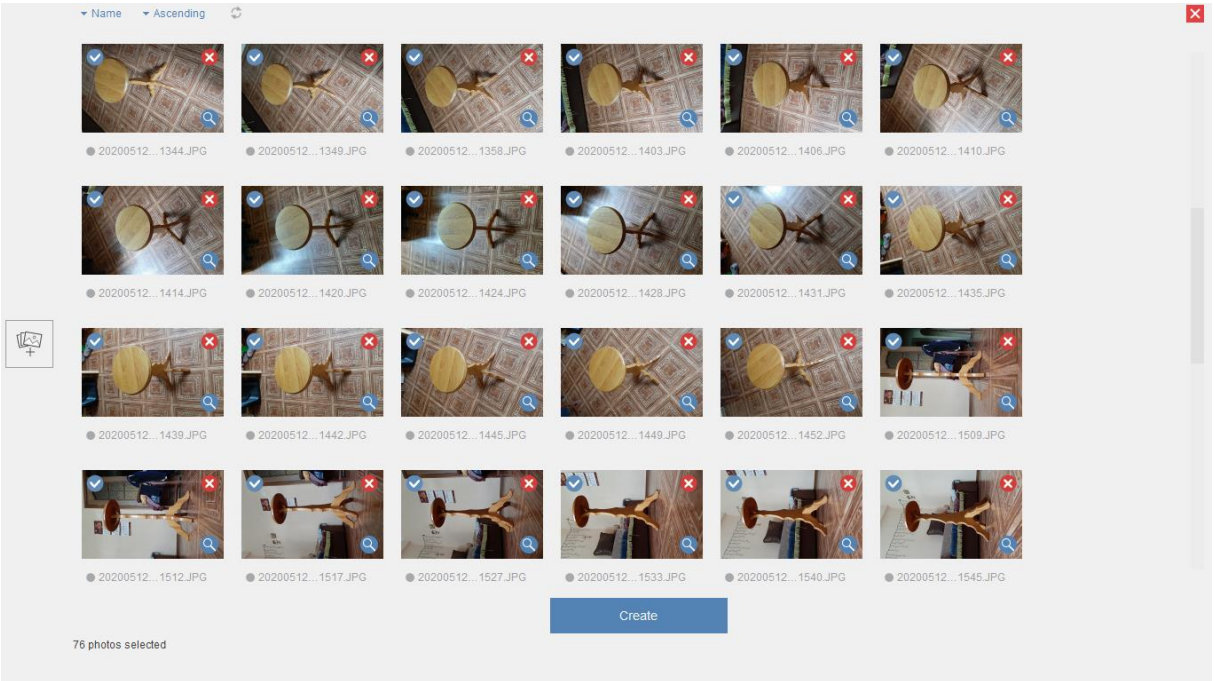

Рисунок 3.2 – Створення проекту Autodesk ReCap

Далі відбувається процес завантаження проекту на сервер (Рис. 3.3).

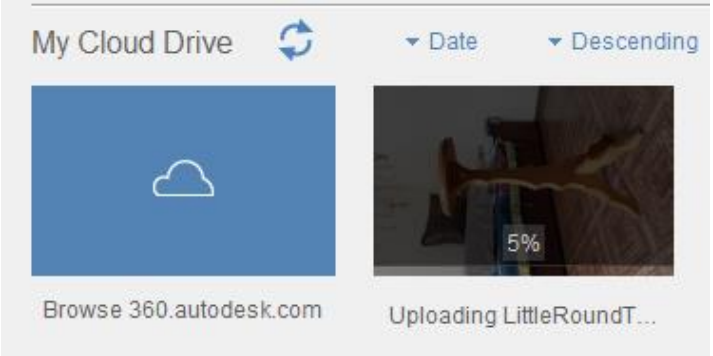

Рисунок 3.3 – Завантаження проекту

Після цього, проводиться обробка фотографій на стороні Autodesk, яка може зайняти значний відрізок часу, оскільки проект може довго знаходитися в черзі, а також потрібен деякий час на обробку, в залежності від складності та об’єму проекту (Рис. 3.4 – 3.5).

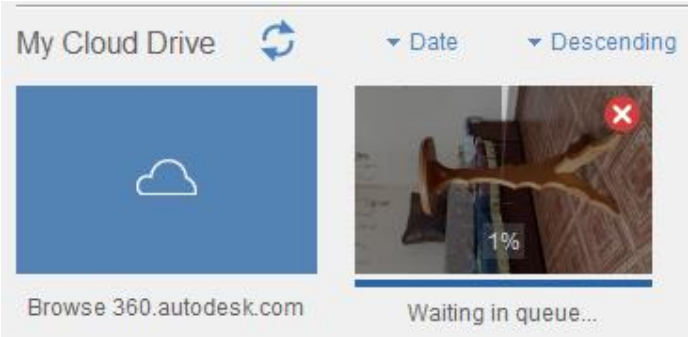

Рисунок 3.4 – Очікування проекту в черзі

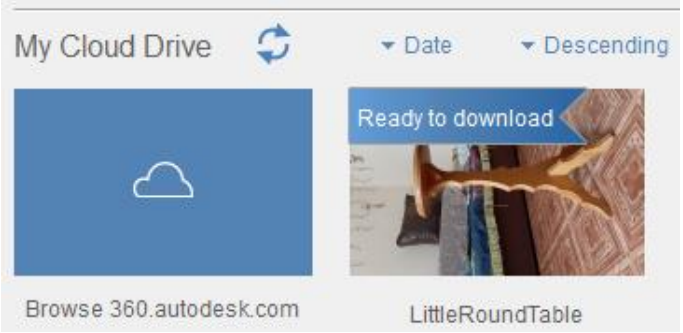

Рисунок 3.5 – Процес обробки проекту

Після того як процес обробки завершується, ми отримуємо триангульовану модель (Рис. 3.6).

Рисунок 3.6 – Результат процесу обробки фотографій

Далі задача полягає в редагуванні моделі, необхідно прибрати зайві елементи, залишивши лише бажаний об’єкт та встановивши реальні розміри об’єкту. Результат редагування моделі показано на рис. 3.7.

Рисунок 3.7 – Результат редагування моделі

Отримати файл моделі у форматі, сумісному із Unity, можна за допомогою функції експорту у формат .fbx (Рис. 3.8).

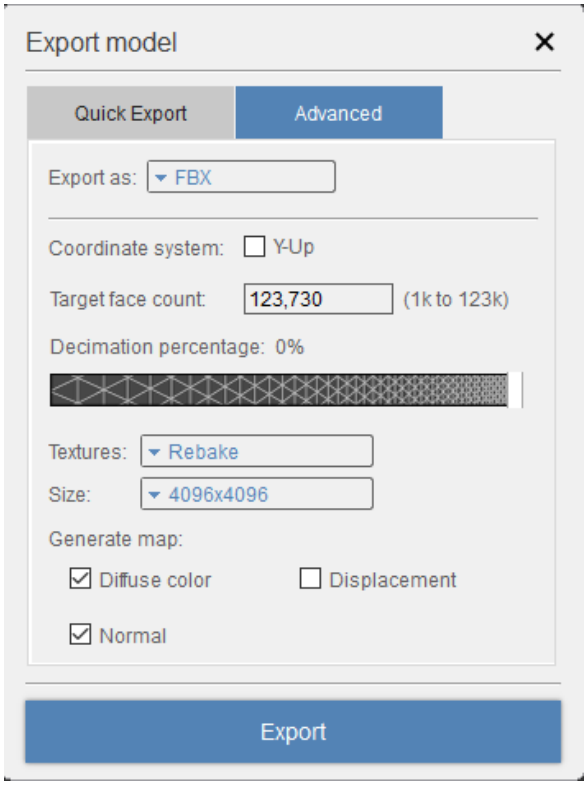

Рисунок 3.8 – Експорт моделі

3.2 Реалізація інтерфейсу програмного додатку

На початку роботи над дизайном додатку було проаналізовано тенденції поточного року та тренди в сучасному світі дизайну. Тому було обрано синій, колір 2020 року за версією Pantone, основним кольором додатку. Також, спираючись на психологію сприйняття, синій колір надає відчуття надійності та впевненості в продукті, крім того асоціюється зі спокоєм, затишком, комфортом, що ідеально підходить для проектування інтер'єрів. У створенні логотипу, який представлено на рис. 3.9, також задіяно ботанічний тропічний тренд, який одночасно надає відчуття задоволення та заспокоєння.

Рисунок 3.9 – Логотип додатку

На етапі розробки інтерфейсу було вирішено, що основною буде сторінка із режимом доповненої реальності (Рис. 3.10).

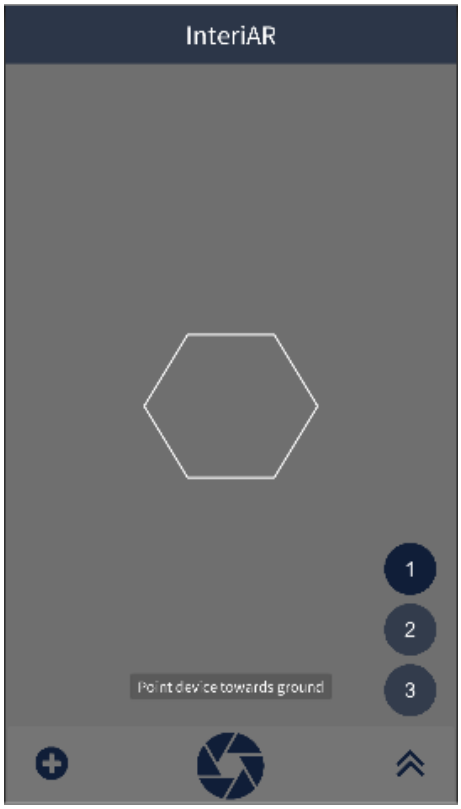

Рисунок 3.10 – Головна сторінка додатку

Інтерфейс головної сторінки додатку містить верхню панель, на якій розміщена назва додатку. Нижня панель сторінки складається із кнопки переходу до меню вибору меблів зліва, кнопки, яка дозволяє розкривати та приховувати список обраних

меблів справа. Список складають кнопки, що дозволяють користувачу обрати певний предмет меблів, для встановлення в режимі доповненої реальності.

Центральна кнопка нижньої панелі для отримання скріншоту екрану та його поширення. Також, головна сторінка доповнена спливаючими повідомленнями, які відображають підказки для користувача.

Сторінка меню вибору меблів, яку представлено на рис. 3.11, складається з групи випадаючих списків. Дане меню відображає групу категорій меблів та предмети меблів кожної категорії у випадаючому списку.

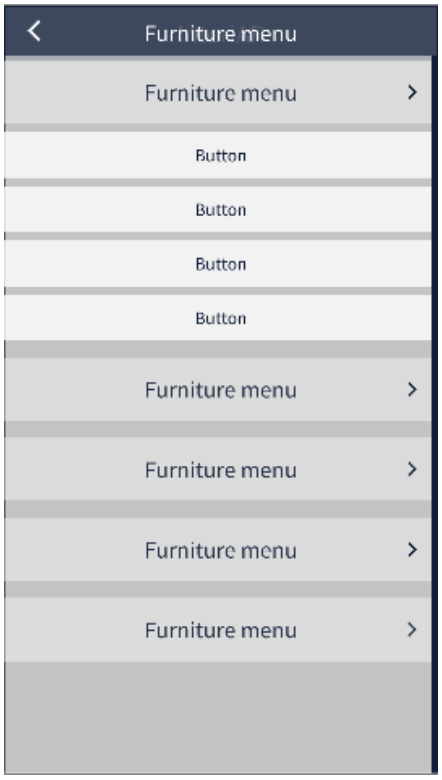

Рисунок 3.11 – Сторінка меню додатку

Реалізація розробленого інтерфейсу відбувалася за допомогою редактора Unity із використанням елементів для створення інтерфейсу. Основними елементами, що використовувались під час розробки є Canvas, Button, Toggle, ListView, ScrollView, Image, Text.

Будь-який інтерфейс сучасного додатку повинен бути адаптивним під різні формати екранів мобільних пристроїв. В Unity це досягається завдяки компонуванню

елементів інтерфейсу таким чином, щоб місця їх розташування були зв’язані з відповідними кутами або сторонами на екрані, або ж встановити масштабування для контролю відповідності розмірів елементів розмірам екрану. Для перевірки адаптивності можна змінювати розширення екрану на панелі Game редактора Unity (Рис. 3.12).

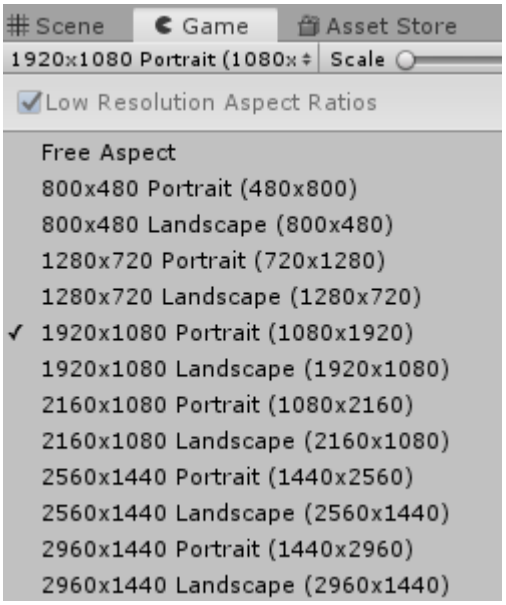

Рисунок 3.12 – Перевірка адаптивності інтерфейсу

Для налаштування взаємодії елементів інтерфейсу використовуються спеціальні скрипти Unity. Скриптинг говорить нашим об'єктам, як себе вести; ігровий процес створюють скрипти і компоненти, прикріплені до об’єктів сцени, а також їх взаємодія один з одним.

Для надання елементам інтерфейсу необхідної реакції на певні події, до потрібних елементів додавалися обробники, які реагують на відповідні події, такі як, наприклад, натискання кнопки. Обробник події OnClick() однієї із кнопок, який визначено в редакторі, представлено на рис. 3.13.

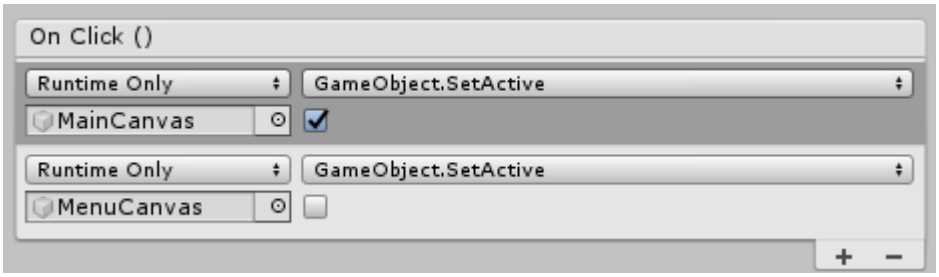

Рисунок 3.13 – Обробник кнопки «Назад» меню вибору меблів

3.3 Програмна реалізація

Пакет Vuforia SDK є набором інструментів розробки програмного забезпечення доповненої реальності та розширеної реальності для різних апаратних засобів і дозволяє реалізувати можливості доповненої реальності в додатку, створюваному з Unity.

Для реалізації взаємодії 3D моделей з оточенням, було використано вбудовані в пакет компоненти та об’єкти, які додаються до сцени в редакторі Unity. В процесі знайомства з можливостями, які надає Vuforia, було вирішено реалізувати розміщення об’єктів із реакцію на поверхню, а не на таргет(зображення, мітку).

Vuforia Ground Plane дозволяє розміщувати цифрові дані на горизонтальних поверхнях в існуючому оточенні, таких як підлога чи стільниця. Підтримується виявлення та відстеження горизонтальних поверхонь, також дозволяє розміщувти об’єкти в повітрі.

Розробка відбувалася починаючи з додавання об’єкта ARCamera (Рис. 3.14). Це особливий тип камери, який підтримує додатки доповненої реальності як для портативних пристроїв, так і для цифрових окулярів.

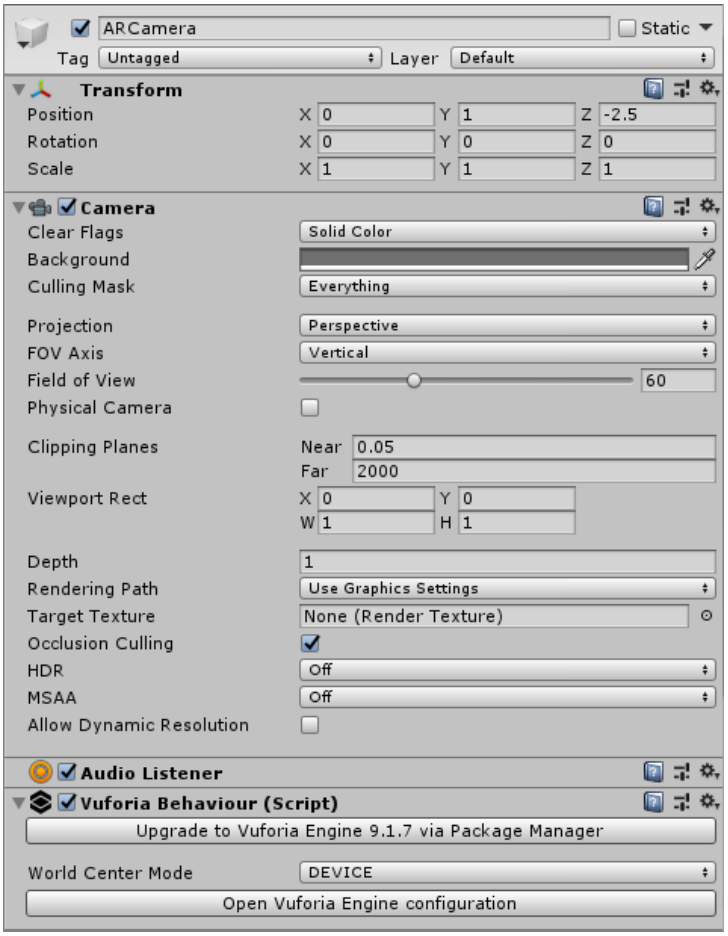

Рисунок 3.14 – Об’єкт ARCamera

Наступним до сцени додається об’єкт Plane Finder (Рис. 3.15), який виконує наступні дії:

- Відслідковує вхідні дані від користувача (наприклад, дотик на екрані пристрою) за допомогою компонента Anchor Input Listener Behaviour.
- Виконує пошук відповідної площини для розміщення вмісту в реальному світі за допомогою компонента Plane Finder Behaviour.
- Розміщує об’єкт у реальному світі за допомогою компонента Content Positioning Behaviour.

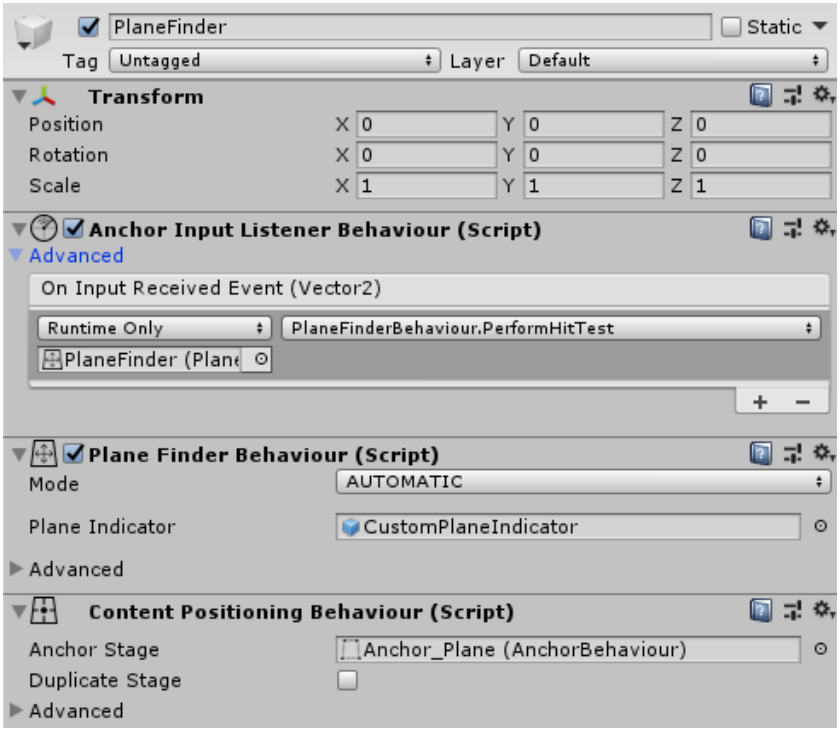

Рисунок 3.15 – Об’єкт Plane Finder

Для розміщення контенту необхідним є батьківський GameObject із компонентом Anchor Behaviour, всередину якого вкладається об’єкт моделі для відображення. Об’єкт із даним компонентом необхідно передавати до Plane Finder як параметр Anchor Stage компонента Content Positioning Behaviour.

Оскільки додаток не передбачає заздалегідь налаштованих для відображення моделей, є необхідність додавати їх під час роботи додатку, тому виникла потреба у створенні такого елементу, як Prefab.

Система Prefab у Unity дозволяє створювати, конфігурувати та зберігати GameObject у комплекті з усіма його компонентами, значеннями властивостей та дочірніми GameObjects, як багаторазовий Asset. Він діє як шаблон, з якого можна створювати нові екземпляри даних у сцені.

Створений Prefab із компонентом Anchor Behaviour наведено на рис. 3.16.

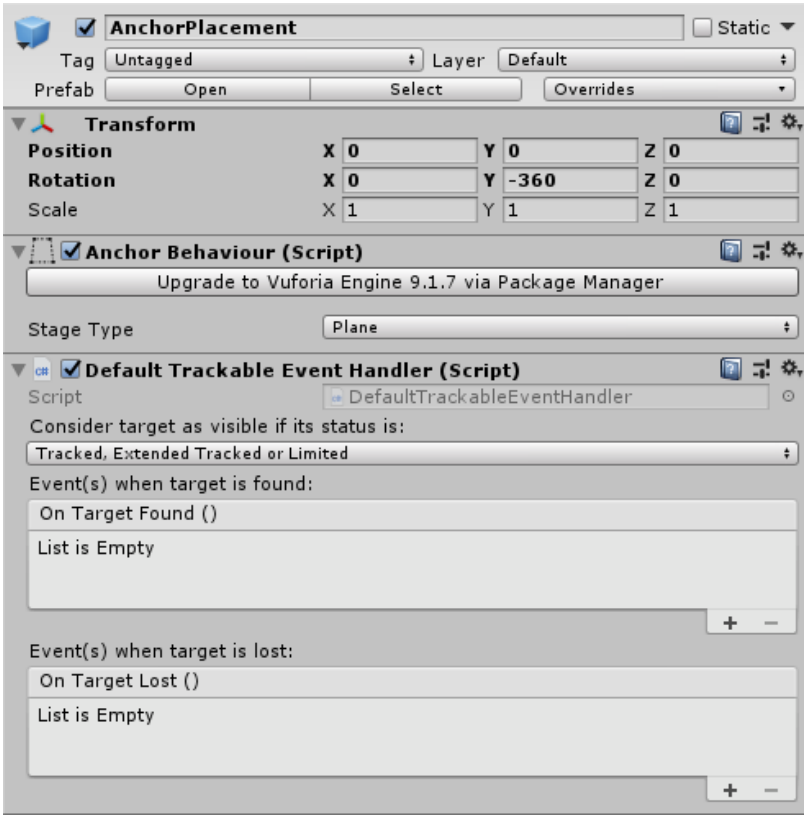

Рисунок 3.16 – Anchor Placement Prefab

Крім необхідних об’єктів Vuforia Engine, створювалися додаткові скрипти для управління розташуванням та положенням моделі в просторі (Рис. 3.17), зміною позиції та повороту моделі (Рис. 3.18). Лістинг створюваних скриптів наведено у Додатку В.

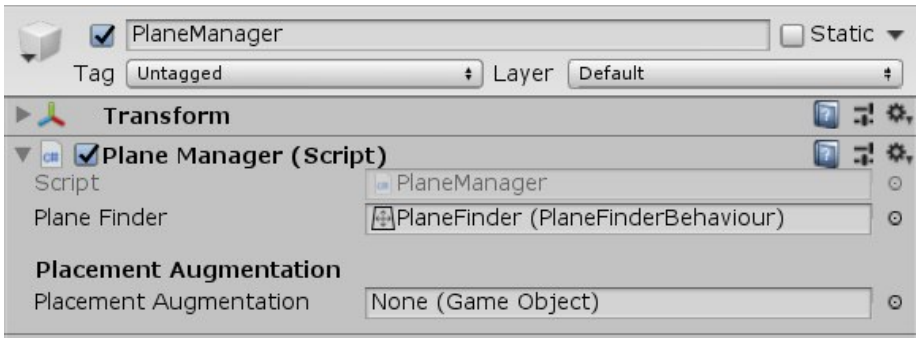

Рисунок 3.17 – Об’єкт Plane Manager

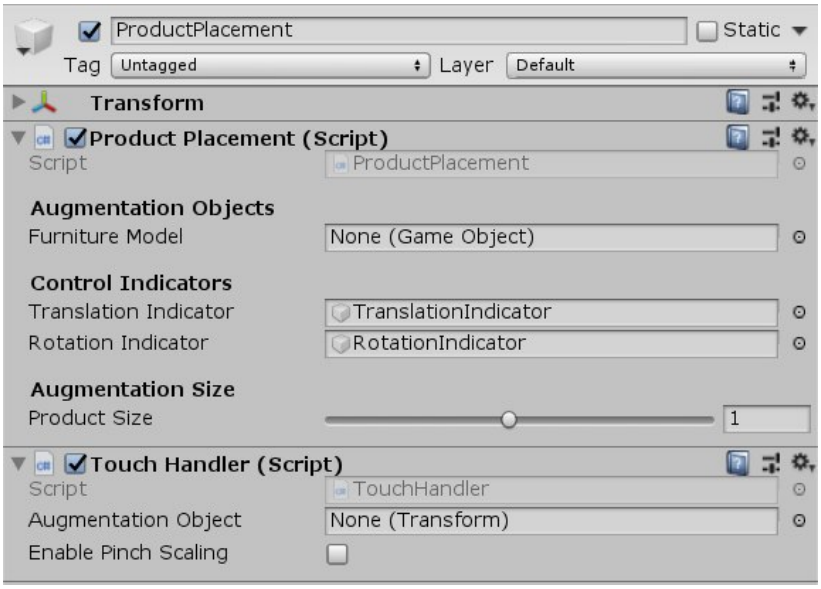

Рисунок 3.18 – Об’єкт Product Placement

Управління частиною інтерфейсу, що пов’язаний із розміщенням моделей в режимі доповненої реальності також організовано за допомогою скрипта, доданий до об’єкта сцени компонент представлено на рис. 3.19. Він відповідає за надання інструкцій користувачу у вигляді спливаючих повідомлень, відображення мітки на поверхні та створення екземплярів кнопок для кожної обраної моделі під час роботи додатку.

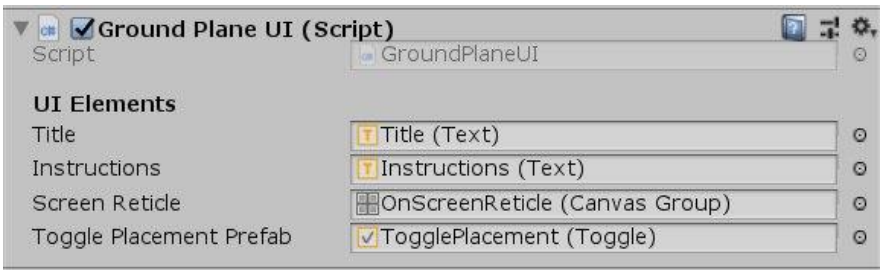

Рисунок 3.19 – Компонент Ground Plane UI

Екземпляр кнопки також потребував створення у вигляді Prefab (Рис. 3.20). Крім необхідних компонентів Unity, шаблон містить скрипт Pointer Handler, який дозволяє побачити назву моделі, а також видалити елемент зі списку.

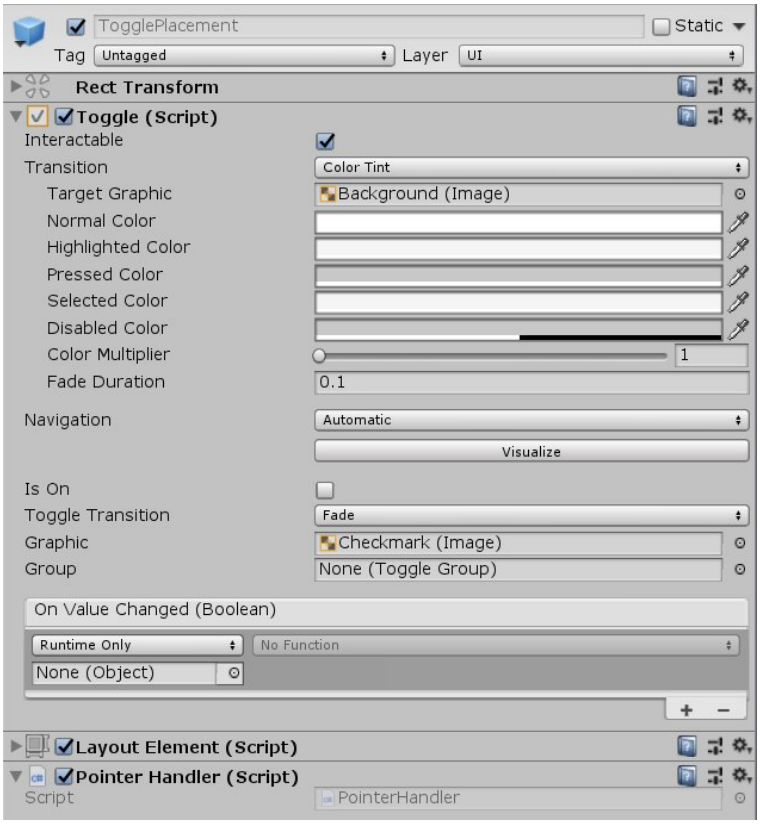

Рисунок 3.20 – Toggle Placement Prefab

При формуванні меню вибору меблів задіяно компонент ListView, який додано до елементу інтерфейсу ScrollView (Рис. 3.21). Він отримує список категорій та список усіх елементів меблів і створює екземпляри кнопок з відповідних префабів (Рис 3.22 – 3.23). Процес завантаження необхідної моделі виконується компонентом RuntimeLoader.

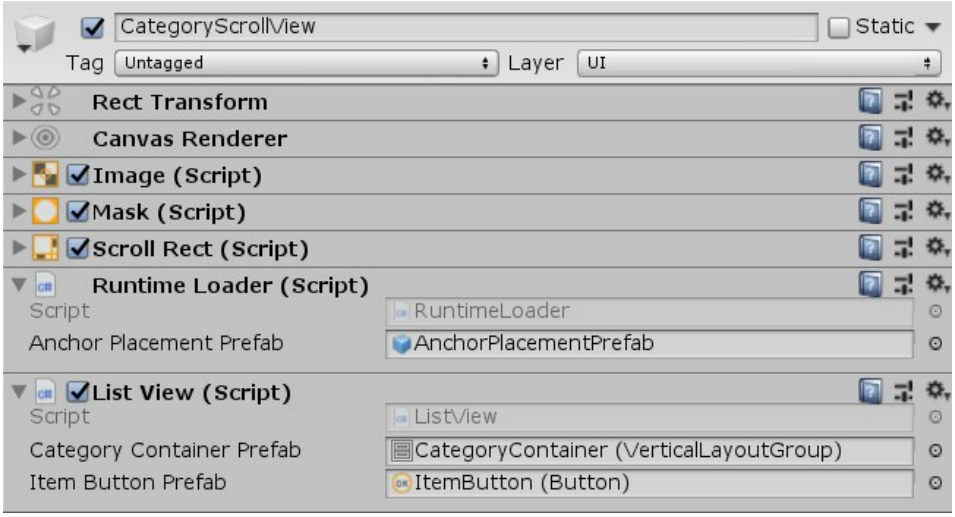

Рисунок 3.21 – Елемент інтерфейсу ScrollView

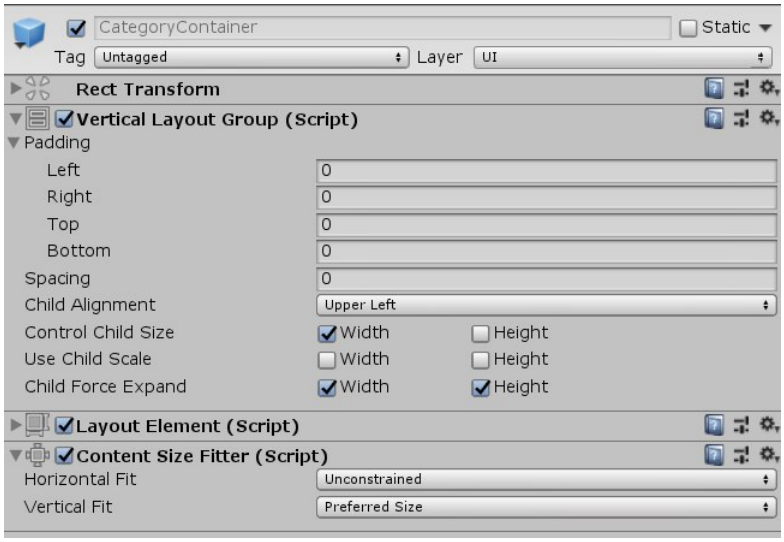

Рисунок 3.22 – Category Container Prefab

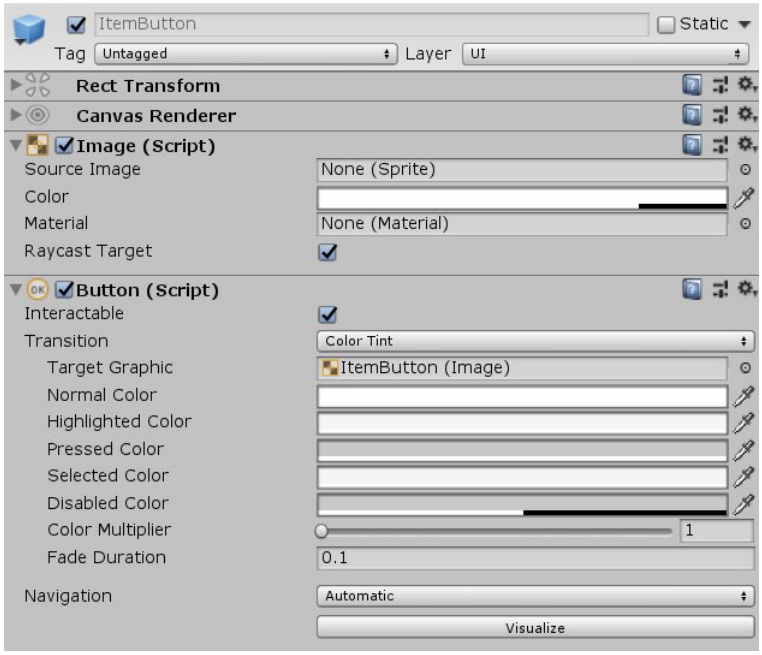

Рисунок 3.23 – Item Button Prefab

В якості платформи для розміщення моделей меблів було обрано Firebase Google Cloud Storage. Під час пошуку хмарного рішення, дана платформа здалася легкою для застосування. Цей висновок не виявився помилковим, за декілька нескладних кроків, було отримано хмарне сховище для мобільного додатку та пакет із Firebase SDK для Unity. Потрібно було зареєструвати додаток в Firebase (Рис. 3.24), додати файл конфігурації до проекту (Рис.3.25) та додати Firebase SDK до пакетів додатку.

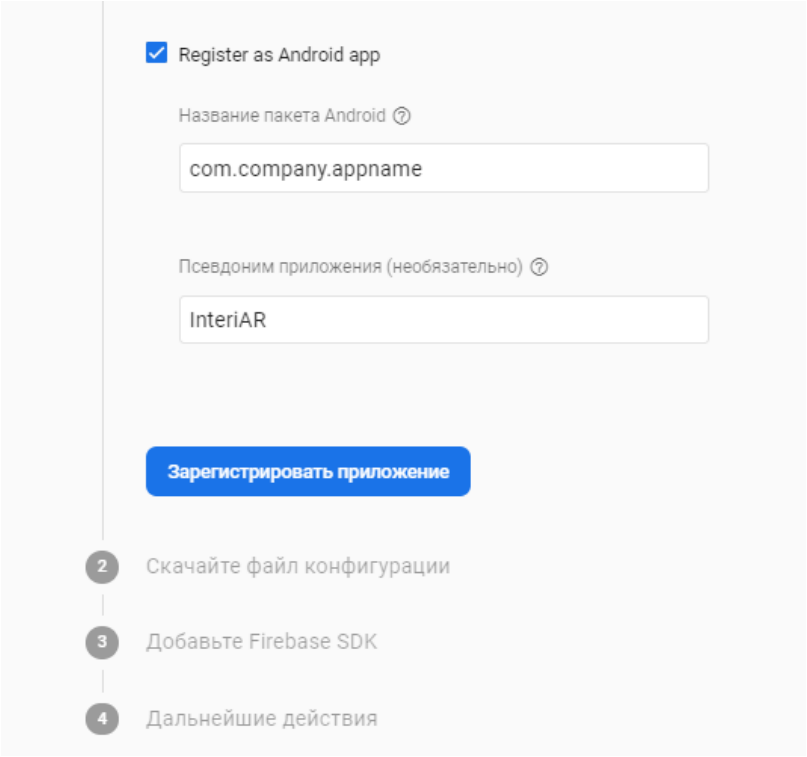

Рисунок 3.24 – Додавання Firebase до Unity додатку

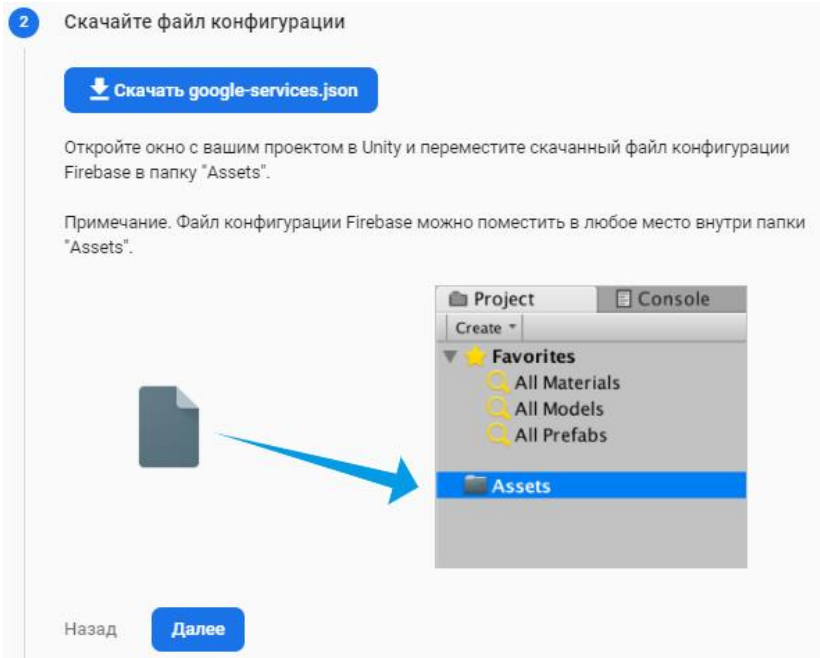

Рисунок 3.25 – Додавання Firebase до Unity додатку

Дані, завантажені у сховище, зберігаються в Asset Bundles. AssetBundle це архівний файл, який містить специфічні для платформи асети, що не містять коду

(наприклад, моделі, текстури, prefabs, аудіо кліпи, і навіть цілі сцени), які Unity може завантажувати під час виконання.

Таке розміщення моделей дозволяє команді проекту змінювати та додавати бажані об’єкти не втручаючись в код додатку.

Вигляд сторінки хмарного сховища Firebase із завантаженими до нього даними представлено на рис. 3.26.

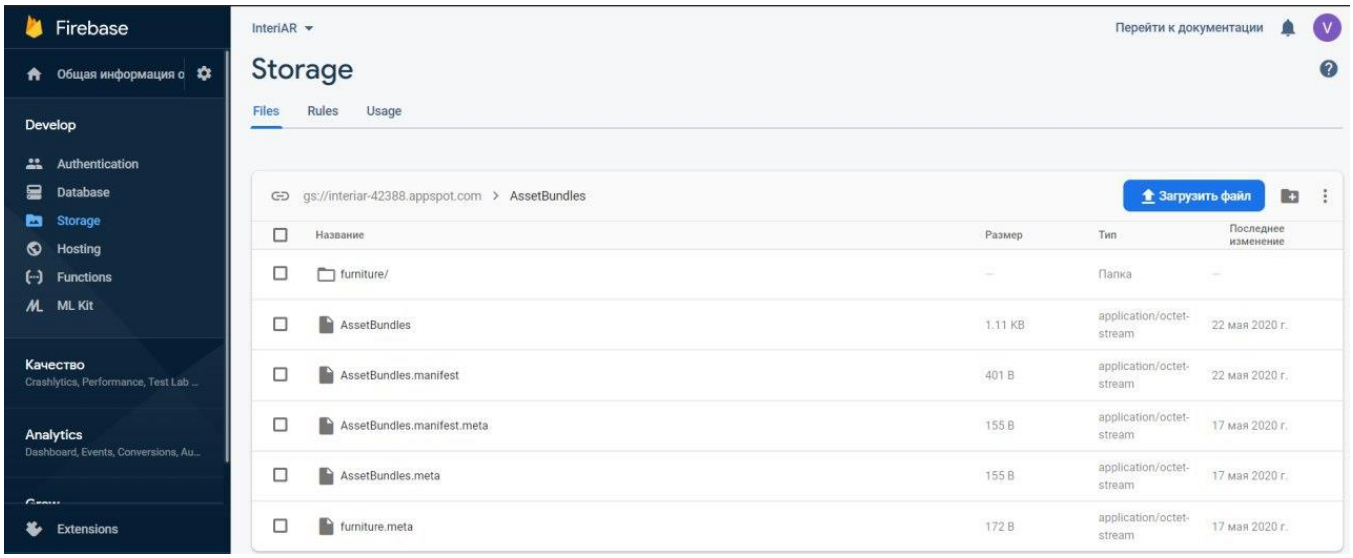

Рисунок 3.26 – Firebase Storage

Процеси, які потребують очікування було вирішено візуалізувати за допомогою прогрес-бару. Поведінку даного елемента описано за допомогою компонента ProgressBar (Рис. 3.27).

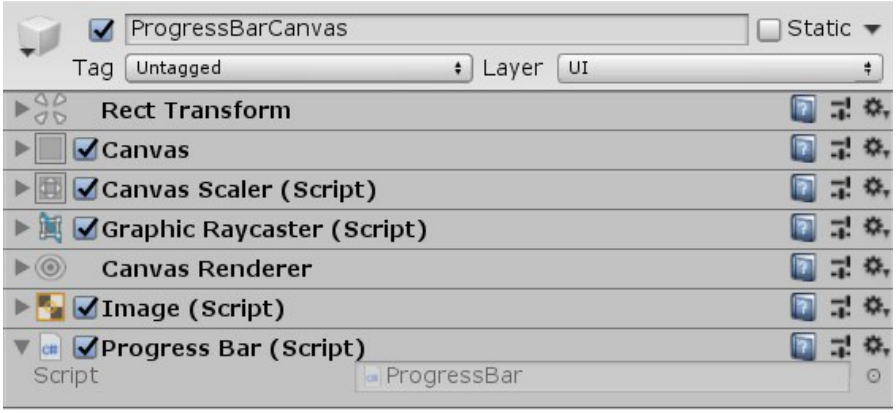

Рисунок 3.27 – Об’єкт Progress Bar

3.4 Використання програмного додатку

Використання програмного додатку передбачає взаємодію користувача із додатком через простий, інтуїтивно зрозумілий інтерфейс. Застосовуючи додаток для планування інтер’єру, він допоможе користувачам уникати покупки меблів, які в результаті можуть виявитися недостатньо підходящими для створеного інтер’єру.

На рис. 3.28 – 3.39 представлено приклад взаємодії користувача із додатком.

Рисунок 3.28 – Splash screen

Після екрану заставки, перш за все користувач бачить головне вікно додатку. За наявною підказкою, яка вказує на кнопку меню, користувач має відкрити меню вибору меблів.

Рисунок 3.29 – Головне вікно

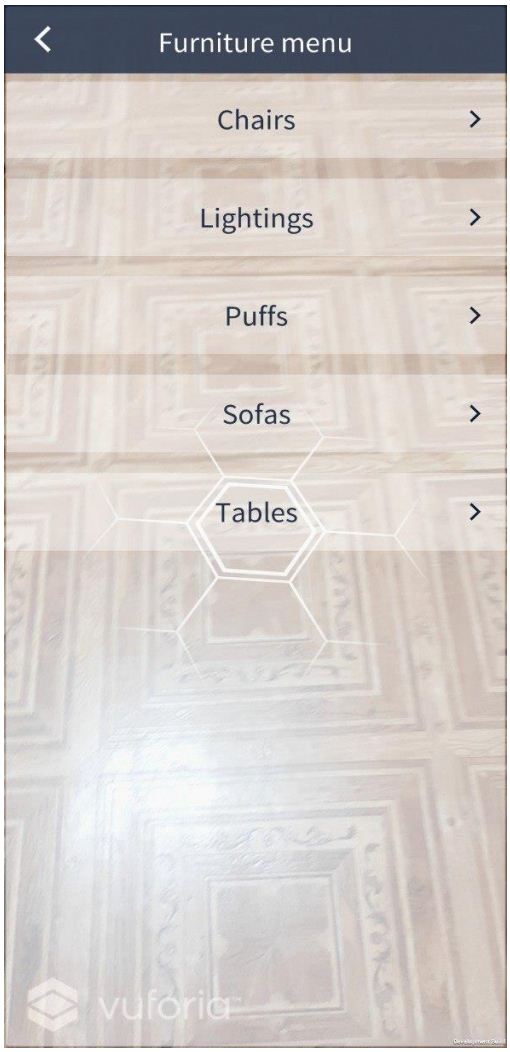

Рисунок 3.30 – Меню вибору меблів

При натисканні на будь-який елемент списку меню, модель предмету меблів буде завантажено. Під час цього процесу користувачу відображається прогрес-бар. Після завершення даного процесу користувач може обирати далі. Завершивши вибір, користувач повертається до головного меню.

Для встановлення предмету меблів в інтер’єрі, згідно із підказками, необхідно розмістити пристрій, направляючи камеру до сканованої поверхні.

Розміщується модель, яку на той момент обрано зі списку, що розгортається з правого боку головного вікна.

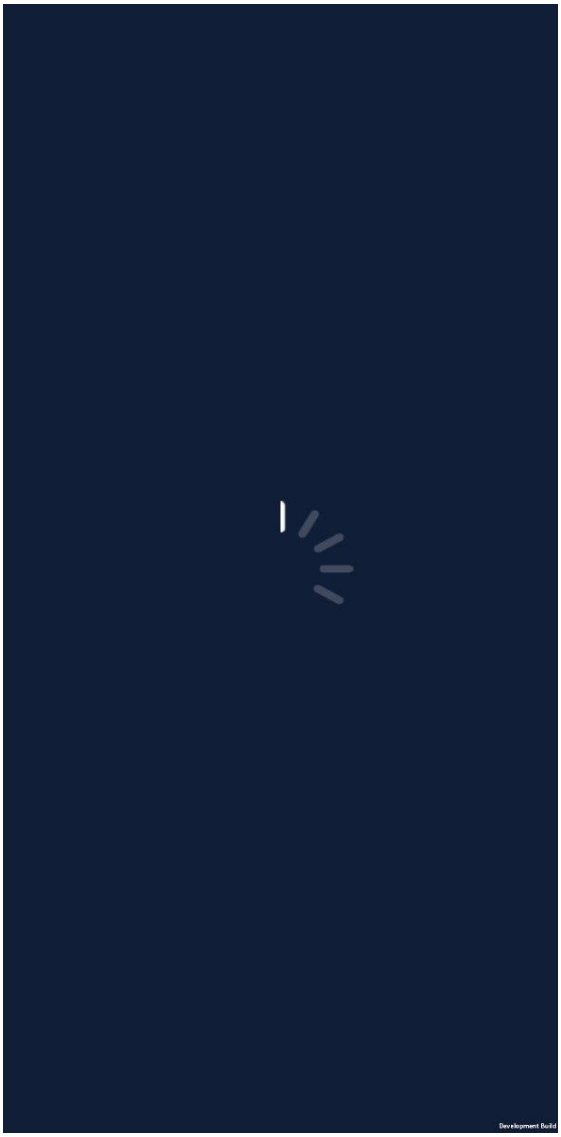

Рисунок 3.31 – Прогрес-бар

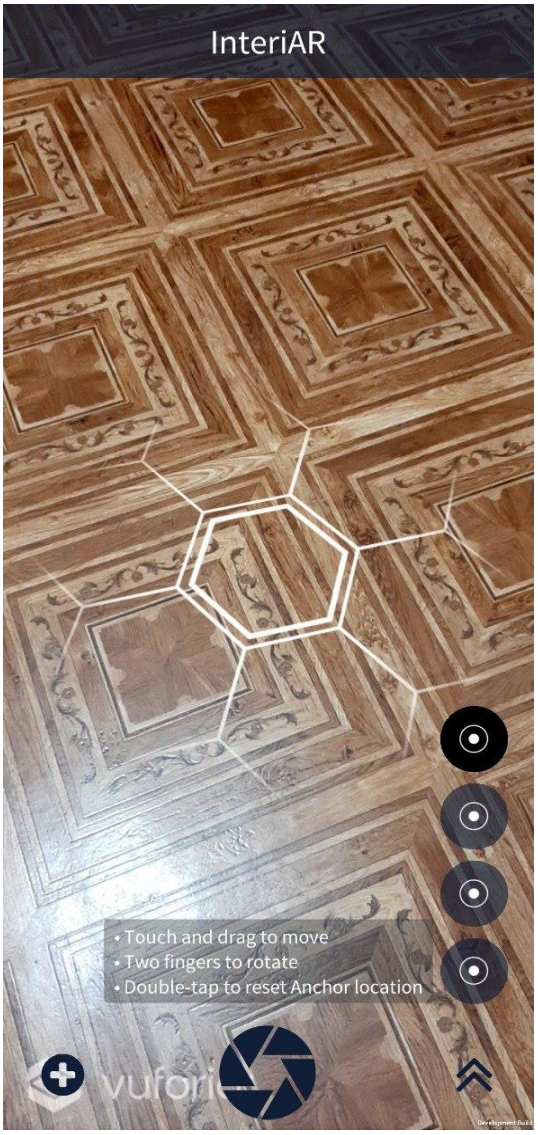

Рисунок 3.32 – Сканування поверхні

Розмістивши предмет меблів на поверхні, користувач має можливість зміни позиції та повороту.

Рисунок 3.33 – Зміна позиції моделі

Рисунок 3.34 – Зміна повороту моделі

У випадку, якщо користувач хоче видалити об’єкт зі сцени, йому необхідно затиснути кнопку зі списку. З’являється назва та кнопка видалення.

При натисканні на центральну кнопку, користувач отримує скріншот результату розміщення меблів та отримує можливість поділитися результатом, відправивши через соцмережі, використовуючи вбудований інстурмент обміну файлами Android, який викликається нашим додатком.

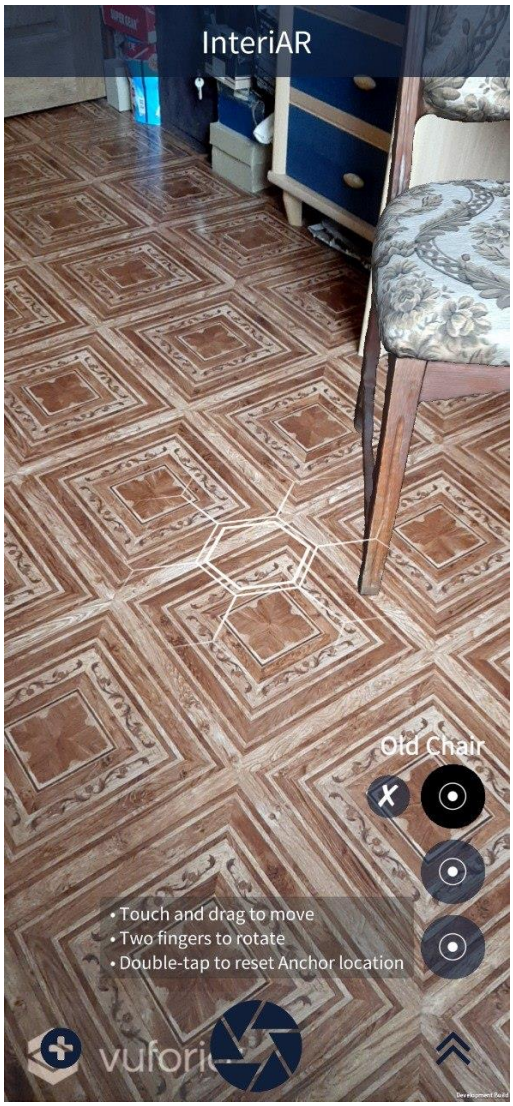

Рисунок 3.35 – Видалення обраного елементу зі списку

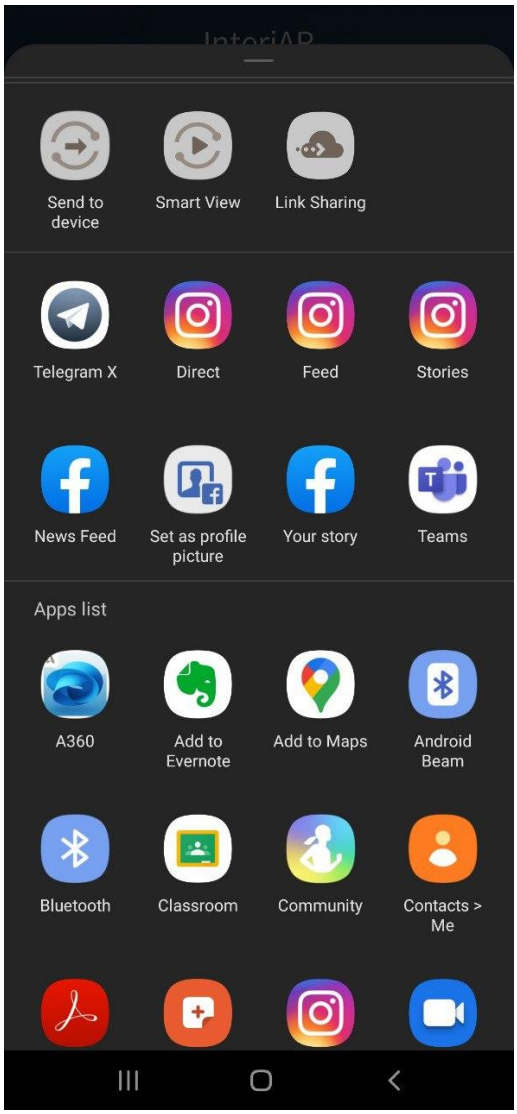

Рисунок 3.36 – Android Share Tool

Для демонстрації, на фото показано реальний об’єкт меблів, який було оброблено за допомогою фотограмметрії, поруч із 3D моделлю, яку було отримано в результаті роботи Autodesk Recap.

Рисунок 3.37 – Реальний об’єкт поруч із 3D моделлю

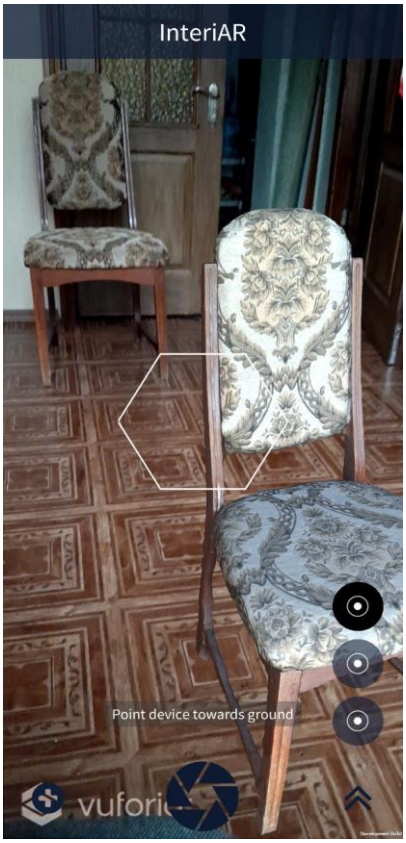

Рисунок 3.38 – Реальний об’єкт поруч із 3D моделлю

Також, можна побачити дану модель в інтер’єрі.

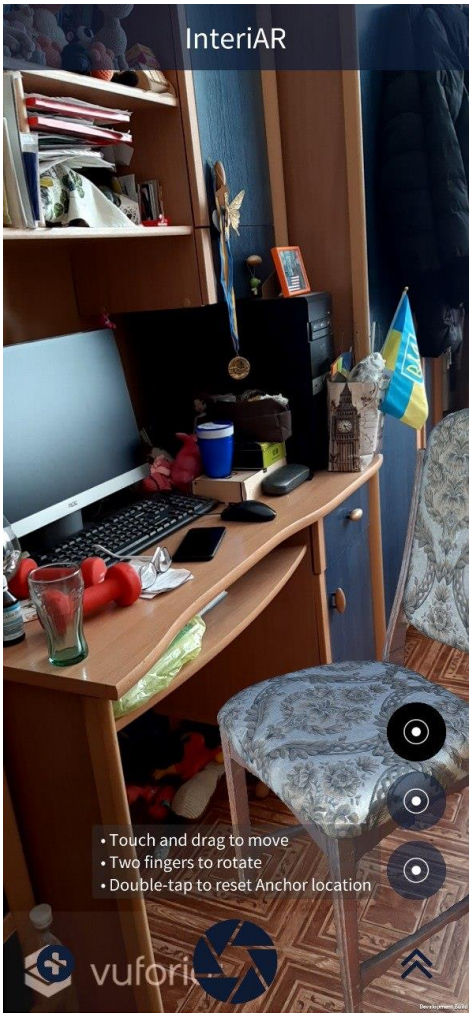

Рисунок 3.39 – 3D модель в інтер’єрі

ВИСНОВКИ

Під час роботи над дипломним проектом було розглянуто предметну область представленого проекту, досліджено актуальність проблеми, проведено аналіз існуючих аналогів. Насамперед було проведено аналіз предметної області, визначено її характеристики, існуючі проблеми, алгоритми та інше.

Було проведено пошук та аналіз вже існуючих аналогічних рішень подібних проблем. Крім того, було детально досліджено питання актуальності проблеми.

Також було сформовано технічне завдання на розробку продукту проекту, в якому зазначені усі визначені вимоги до додатку.

Крім того, частину розділу було присвячено встановленню мети та задач проекту, а також вибору засобів розробки продукту проекту, що будуть використані для досягнення мети проекту та плануванню робіт.

У другому розділі було проведено моделювання інформаційної системи додатку доповненої реальності для облаштування інтер’єрів меблями, представлене у вигляді Use case та IDEF.

Третій розділ присвячено детальному опису процесу розробки набору 3D моделей меблів, розробки інтерфейсу та програмної реалізації додатку.

Результатом роботи є проведений аналіз предметної області, сформована мета та вимоги проекту, а також проведене моделювання програмного продукту та виконано планування робіт. В результаті розробки, згідно із моделюванням було отримано мобільний додаток доповненої реальності.

Мета даної роботи, полягала у розробці додатку доповненої реальності для облаштування інтер’єрів меблями. Основними задачами для досягнення даної мети які було поставлено на початку виконання розробки продукту проекту були:

- детальний аналіз предметної області та аналогічних продуктів;
- вибір технологій розробки продукту;
- розробка набору моделей меблів;
- розробка внутрішнього модулю додатку;

– розробка інтерфейсу додатку;
– проведення тестування продукту.

Засобами Autodesk Recap було реалізовано 3D моделі додатку, а за допомогою Unity та Vuforia було розроблено інтерфейс та функціонал додатку. В ході виконання роботи було повністю виконано задачі проекту.

Розроблений програмний продукт відповідає встановленим функціональним вимогам, а саме, реалізовано функціонал вибору меблів зі списку отриманих із фотографій 3D моделей, розміщення їх у режимі доповненої реальності, зміна позиції та повороту моделі. Крім того, отриманий продукт дозволяє користувачу зберігати результат у вигляді скріншоту та ділитися отриманим результатом.

Також в подальших планах є вдосконалення продукту проекту, що може бути оформлено в нову наукову роботу.

Робота була представлена на конференції IMA-2020, де зайняла перше місце. Сторінку зі збірника тез наведено у Додатку Г.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Доповнена реальність, або AR-технології. Як це працює? [Електронний ресурс] – Режим доступу до ресурсу: http://thefuture.news/page1837780.html.

2 Доповнена реальність (AR) [Електронний ресурс] – Режим доступу до ресурсу: https://www.it.ua/knowledge-base/technology-innovation/dopolnennaja-realnost-ar.

3 7 приложений для смартфона, которые в ... - Novate.RU [Електронний ресурс] – Режим доступу до ресурсу: https://novate.ru/blogs/230618/46785/.

4 IKEA Place [Електронний ресурс] – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=com.inter_ikea.place&hl=uk.

5 AR-приложение IKEA Place для подбора мебели [Електронний ресурс] – Режим доступу до ресурсу: https://itc.ua/news/ar-prilozhenie-ikea-place-dlya-podbora-mebeli-vyishlo-na-android/.

6 Housecraft [Електронний ресурс] – Режим доступу до ресурсу: https://www.housecraftapp.com/.

7 Myty [Електронний ресурс] – Режим доступу до ресурсу: https://myty.app/en/app.

8 Best AR SDK for development for iOS and Android [Електронний ресурс] – Режим доступу до ресурсу: https://thinkmobiles.com/blog/best-ar-sdk-review/.

9 Vuforia Engine Library [Електронний ресурс] – Режим доступу до ресурсу: https://library.vuforia.com/getting-started/overview.html.

10 ReCap | Reality Capture Software | 3D Scanning ... - Autodesk [Електронний ресурс] – Режим доступу до ресурсу: https://www.autodesk.com/products/recap/overview.

11 Unity Scripting Reference (2019.2) [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unity3d.com/2019.2/Documentation/ScriptReference/index.html.

12 Scripting in Unity for experienced programmers [Електронний ресурс] – Режим доступу до ресурсу: https://unity.com/how-to/programming-unity.

13 Unity Game Development Languages ... - MakeUseOf [Електронний ресурс] – Режим доступу до ресурсу: https://www.makeuseof.com/tag/unity-game-development-languages/.

14 Проектирование информационных систем [Електронний ресурс] – Режим доступу до ресурсу: https://sites.google.com/site/anisimovkhv/learning/pris.

15 How to take the right photos to be used for ReCap Photo [Електронний ресурс] – Режим доступу до ресурсу: https://knowledge.autodesk.com/support/recap/learn-explore/caas/sfdcarticles/sfdcarticles/How-to-take-the-right-photos-to-be-used-for-ReCap-360.html.

16 Firebase Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/docs?authuser=0.

17 Creating an AssetBundle [Електронний ресурс] – Режим доступу до ресурсу: https://learn.unity.com/tutorial/introduction-to-asset-bundles?signup=true#5ce589b4edbc2a106aa7b47d.

18 IMA-2020 /. – Суми, 2020. – (Сумський державний університет). – С. 119.

19 Моделирование на UML. [Електронний ресурс] – Режим доступу до ресурсу: http://.uml3.ru/.

20 A Guide to the Project Management Body of Knowledge (PMBOK® Guide)– Sixth Edition , 2017. – 760 с.

ДОДАТОК А
ТЕХНІЧНЕ ЗАВДАННЯ
на розробку «Додаток доповненої реальності для облаштування інтер'єрів меблями»

1 Призначення й мета створення програмного продукту

1.1 Призначення розробки

Додаток призначений для демонстрації створених на основі ряду знімків 3D моделей меблів в інтер'єрі за допомогою технології доповненої реальності.

1.2 Мета створення додатку –

полягає у розробці додатку доповненої реальності для облаштування інтер'єрів меблями.

1.3 Цільова аудиторія

У цільовій аудиторії програмного продукту можна виділити наступні групи:

1. Клієнти, користувачі додатку, які зацікавлені у виборі меблів.
2. Меблеві магазини, які зацікавлені у збільшенні продажів.
3. Інші зацікавлені користувачі.

2 Вимоги до програмного продукту

Розроблюваний програмний продукт повинен бути реалізований у вигляді android-додатку, (далі – продукт, додаток), клієнтська частина якого є Unity-сцена, з набором елементів керування (кнопки та інші) та забезпечує виконання функціональних можливостей, визначених у пункті 2.3.

2.1 Вимоги до технології розробки

Продукт розробляється ітеративно з урахуванням принципів та технологій уніфікованого процесу розроблення програмного забезпечення. Програма повинна

бути реалізована мовою C # з використанням технології Microsoft.Net. Також повинні бути використані технології фотограмметрії Autodesk ReCap – для створення тривимірних моделей меблів та технології доповненої реальності Vuforia - для відображення моделей.

2.2 Вимоги до програмного забезпечення

Для коректної роботи додатку доповненої реальності необхідні наступні мінімальні характеристики системи:

- Platform Support: Android 4.2 and above;
- CPU Architectures: Android: armv7a, arm64-v8a;
- Працездатна основна камера;

2.3 Вимоги до функціональних характеристик

Додаток повинен забезпечувати виконання наступних функції:

• наявність створених із колекцій фото 3D моделей меблів;

• перегляд списку моделей меблів;

• перегляд моделі в додатку;

• перехід до режиму доповненої реальності;

• редагування позиції моделі;

• додавання декількох моделей до сцени;

3 Склад і зміст робіт зі створення додатку

Основні етапи розробки повинні складатися з наступних пунктів:

• Оформлення завдання для дипломної роботи;

• Планування роботи. Розроблення ТЗ ,побудова мережевого графіку та діаграми Ганта;

• Розробка набору моделей меблів;

• Розробка внутрішнього модулю додатку;

• Розробка інтерфейсу додатку;

• Провести тестування програмного продукту

• Розроблення інструкції користувача;

• Оформлення пояснювальної записки;

• Здача пояснювальної записки;

• Презентація роботи та її захист.

4 Вимоги до складу й змісту робіт із введення додатку в експлуатацію

В процесі розробки програмного продукту необхідно перевірити готовність продукту до передачі в експлуатацію, а також забезпечити його придатність та відповідність ТЗ.

Додаток передається в експлуатацію шляхом створення *.apk файлу та надання його замовнику із можливістю розміщення на платформі GooglePlay, у разі якщо у замовника наявний сплачений аккаунт на даній платформі.

5 Порядок контролю та приймання

Контроль коректності функціонування та придатності додатку здійснюється замовником на основі наданої пояснювальної записки до дипломної роботи та програмних файлів.

Контроль ходу виконання проекту здійснюється на основі календарного плану виконання дипломної роботи:

• Перевірка завдання дипломної роботи.

• Перевірка ТЗ, мережевого графіка та діаграми Ганта.

• Перевірка структури додатку.

• Перевірка наявності необхідного функціоналу.

• Перевірка коректності тестування.

• Перевірка інструкції користувача.

• Здача ПЗ.

• Презентація.

ДОДАТОК Б

Планування робіт

Мета проекту: розробити додаток доповненої реальності для облаштування інтер'єрів меблями, за допомогою якого користувач зможе легко обрати бажані предмети інтер'єру із наданого переліку, переглянути їх на екрані та розташувати в будь-якій частині свого інтер'єру за допомогою технологій доповненої реальності. Надані користувачам моделі мають відображати реальні об'єкти.

Результати деталізації мети проекту методом SMART представлені у табл.1.

Таблиця 1 – Деталізація мети проекту методом SMART

S - Specific	Розробити додаток доповненої реальності для облаштування інтер'єрів меблями
M - Measurable	Оскільки даний проект є некомерційним та не має оцінювачів. Інструментами для виміру в даному випадку є експертна група оцінювання його роботи.
A - Achievable	Мета даного проекту може вважатися досяжною оскільки вона була ретельно вивчена, обговорена та узгоджена командою проекту.
R - Relevant	Усе необхідне для реалізації проекту апаратне та програмне забезпечення доступне та коректно працює (платформа для розробки в реальному часі - Unity Engine, середовище розробки - Visual Studio та Autodesk ReCap – хмарний сервіс для створення моделей меблів за серіями фото реальних об'єктів). Команда проекту достатньо кваліфікована та обізнана в необхідних для виконання проекту інструментах.
T - Time-framed	Для даного проекту встановлені часові обмеження, які визначені керівником проекту та описані в календарному плані. Проект буде виконано вчасно, що підтверджується календарним планом проекту.

Планування змісту структури робіт IT-проекту (WBS)

WBS служить інструментом для планування змісту структури робіт проекту. Це графічне представлення проекту у вигляді ієрархічної структури робіт, яка формується шляхом розбиття проекту на конкретні результати, що повинні бути досягнуті на певних рівнях для виконання мети. WBS є, безумовно, найбільш ефективним способом для візуалізації всього проекту, а також надає можливість сфокусувати увагу учасників проекту на очікуваному результаті.

Оскільки описаний результат є бажаним досягненням проекту, спланувавши зміст структури робіт, отримаємо достатньо стабільний набір категорій, узгоджених із командою, на які вона може опиратися в майбутньому. Сформована діаграма WBS представлена на рис. Б.1.

Планування структури організації, для впровадження готового проекту (OBS)

Після закінчення побудови WBS виконується розробка організаційної структури виконавців проекту OBS (Organization structure). OBS являє собою графічне представлення проекту у вигляді осіб, що задіяні в реалізації проекту, відповідальних за певні пакети робіт.

Правильно сформована організаційна структура визначає роботу кожного учасника проекту і те, як він вписується в загальну систему. Простіше кажучи, організаційна структура визначає, хто що робить, для досягнення цілей проекту. Кінцева організаційна структура виконавців проекту представлена на рис. Б.2.

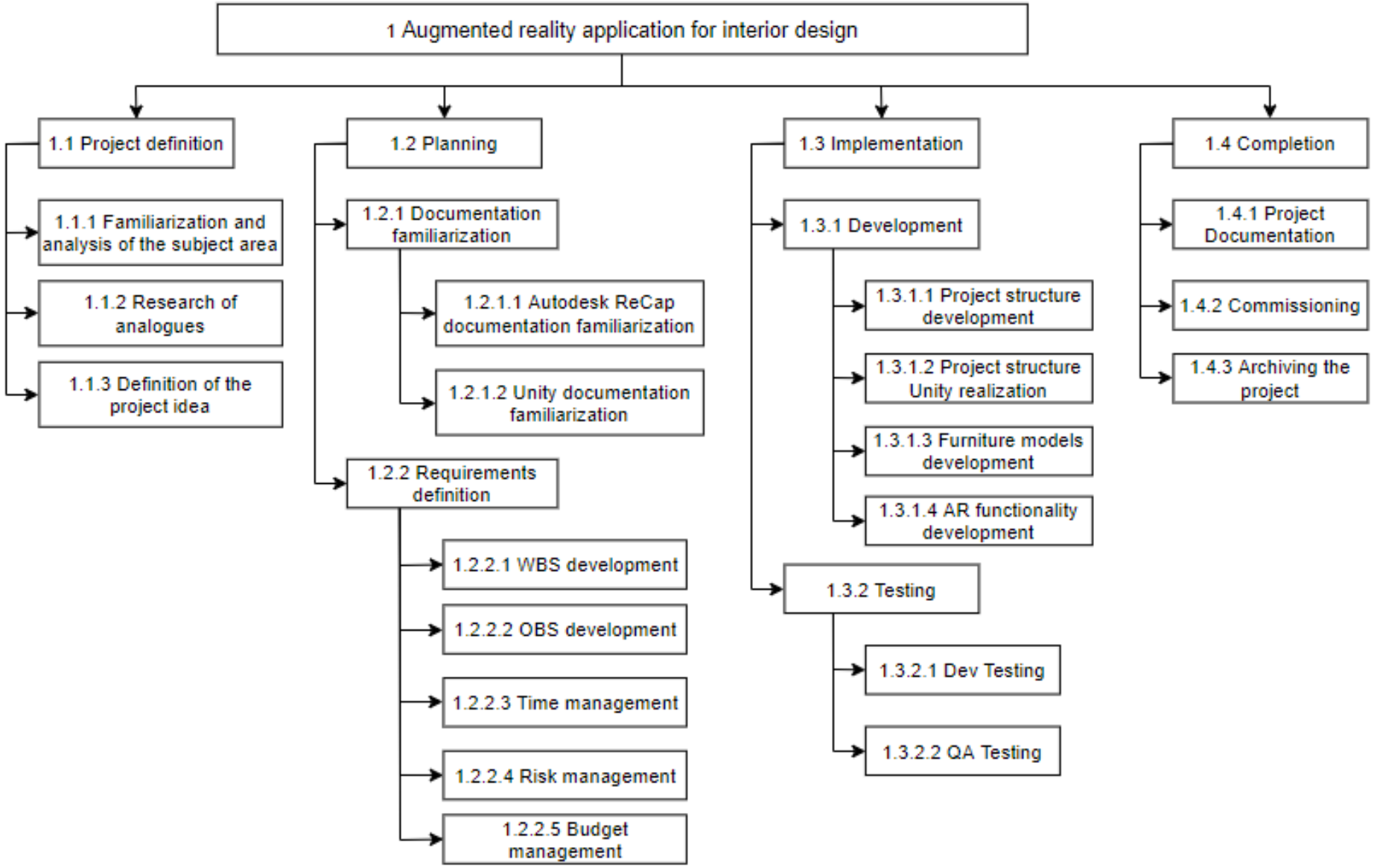

Рисунок Б.1 – Ієрархічна структура робіт проекту

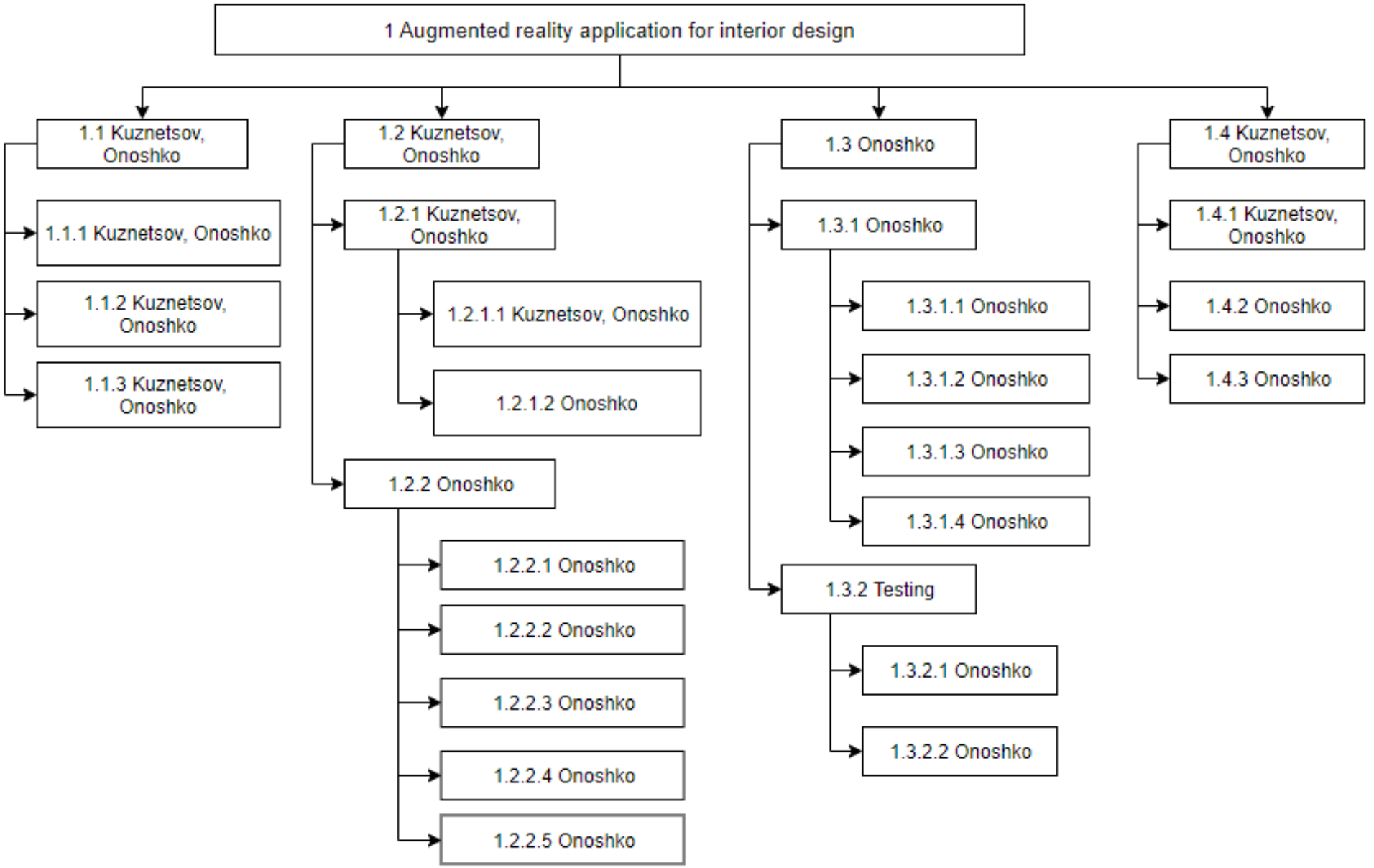

Рисунок Б.2 – Організаційна структура проекту

Побудова матриці відповідальності (виконавців пакетів робіт)

На основі двох попередньо розроблених структур проекту (WBS та OBS) створюється комунікаційний документ, який формалізує відповідальність ролей або конкретних людей за виконання завдань та отримання кінцевих результатів проекту – мариця відповідальності, або Responsibility Assignment Matrix (RAM). Існує декілька моделей матриць відповідальності, які враховують тип втручання в проект. Розроблена матриця відповідальності представлена на рис Б.3.

Модель RACI робить проект більш організованим, вона інформує про навантаження команди, оскільки показує, яку роль покладено на кожного учасника проекту.

- Responsible: той хто виконує завдання, є відповідальним за роботи для досягнення поставленого завдання.
- Accountable: той хто відповідає за правильне виконання на кінцевому етапі поставленого завдання, затверджує виконання тієї чи іншої роботи. Відповідальний за результати та якість.
- Consulted: той з ким встановлена комунікація протягом проекту, йому повідомляють про рішення та обговорюють завдання.
- Informed: той кого необхідно інформувати про процес виконання завдань, з ним встановлена одностороння комунікація.

ID	WBS Element Description	PM	Team Leader	Developer	Analyst	QA
1.1.1	Familiarization and analysis of the subject area	R	A	-	-	-
1.1.2	Research of analogues	C	A/R	I	C	-
1.1.3	Definition of the project idea	C	A/R	C	C	-
1.2.1.1	Autodesk ReCap documentation familiarization	-	A	R	-	-
1.2.1.2	Unity documentation familiarization	-	A	R	-	-
1.2.2.1	WBS creation	R	C	C	A	-
1.2.2.2	OBS creation	R	C	C	A	-
1.2.2.3	Time management	C	A	I	R	-
1.2.2.4	Risk management	C	A/R	C	C	-
1.2.2.5	Budget management	A/C	-	-	R	-
1.3.1.1	Project structure development	R	A	C	C	I
1.3.1.2	Project structure Unity realization	I	A/C	R	-	I
1.3.1.3	Furniture models development	I	A/C	R	-	I
1.3.1.4	AR functionality development	I	A/C	R	-	I
1.3.2.1	Dev Testing	I	A	R	C	I
1.3.2.2	QA Testing	I	A	I	-	R
1.4.1	Project Documentation	R	A	C	C	C
1.4.2	Commissioning	C	A/R	I	I	I
1.4.3	Archiving the project	I	A/R	I	I	I

Рисунок Б.3 – Матриця відповідальності проекту **(**RAM)

Розробка PDM-мережі (розгорнутий вигляд мережевих діаграм Ганта)

Для кожного пакету робіт, визначеного в структурі WBS, створюються мережеві моделі, в яких встановлюють зв’язки між усіма роботами проекту. Такі мережеві моделі дозволяють отримати визначену тривалість виконання проекту, а також окремих пакетів робіт. На даний момент частіше використовуються PDM (Product Data Management)-мережі або мережі типу «вершина-робота», які складаються з двох типів елементів: робіт, які розташовані у вузлах, та стрілок, які представлють собою переходи та є логічними взаємозв’язками між роботами проекту. В даному випадку для розробки PDM-мережі було використано пакет MS Project, результат наведено на рис. Б.4-Б.6.

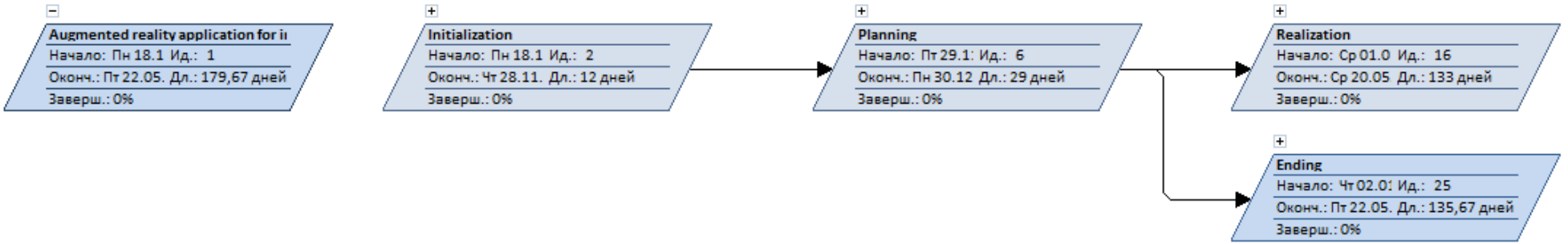

Рисунок Б.4 – Сумарні задачі PDM-мережі

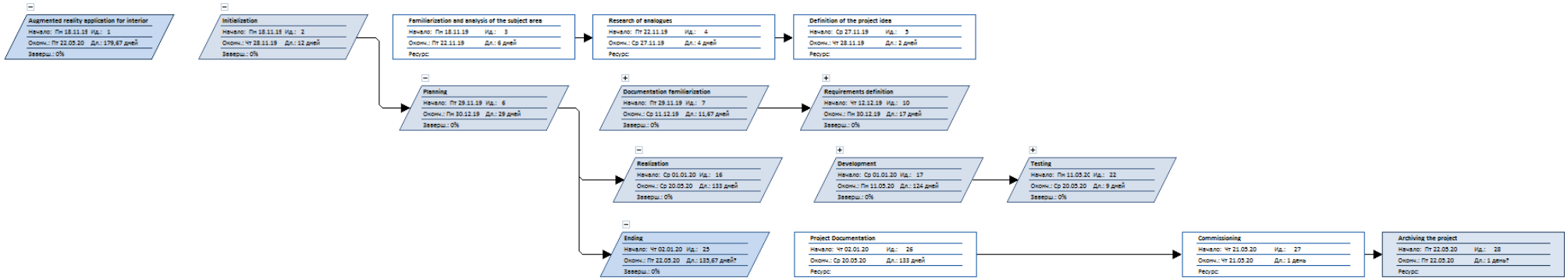

Рисунок Б.5 – PDM-мережа у згорнутому до основних моментів вигляді

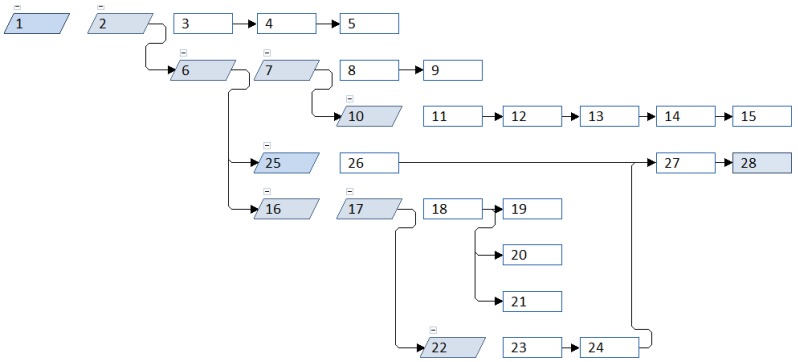

Рисунок Б.6 – PDM-мережа у повному вигляді

Побудова діаграми Ганта

Діаграма Ганта дозволяє отримати реальне уявлення про тривалість проекту з урахуванням вихідних та святкових днів, а також з урахуванням обмеженості ресурсів. Вона являє собою горизонтальні відрізки, розташовані між двома осями: списком робіт та шкалою часу. Розташування даного відрізка позначає початок, кінець та тривалість певної роботи, також дуже зручно можуть бути показані зв'язки між роботами та відсоток виконаних робіт по кожному завданню.

Список робіт діаграми, що спирається на раніше побудовану WBS, представлений на рис. Б.7. Розроблена завдяки пакету MS Project діаграма Ганта представлена на рис. Б.8.

Augmented reality application for interior design	**Пн 18.11.19**	**Пт 22.05.20**
Initialization	**Пн 18.11.19**	**Чт 28.11.19**
Familiarization and analysis of the subject area	Пн 18.11.19	Пт 22.11.19
Research of analogues	Пн 25.11.19	Ср 27.11.19
Definition of the project idea	Чт 28.11.19	Пт 29.11.19
Planning	**Пн 02.12.19**	**Пн 30.12.19**
Documentation familiarization	**Пн 02.12.19**	**Ср 11.12.19**
Autodesk ReCap documentation familiarization	Пн 02.12.19	Чт 05.12.19
Unity documentation familiarization	Пт 06.12.19	Ср 11.12.19
Requirements definition	**Чт 12.12.19**	**Пн 30.12.19**
WBS creation	Чт 12.12.19	Пт 13.12.19
OBS creation	Пн 16.12.19	Вт 17.12.19
Time management	Ср 18.12.19	Пт 20.12.19
Risk management	Пн 23.12.19	Ср 25.12.19
Budget management	Чт 26.12.19	Пн 30.12.19
Realization	**Чт 02.01.20**	**Ср 20.05.20**
Development	**Чт 02.01.20**	**Пн 11.05.20**
Project structure development	Чт 02.01.20	Пт 17.01.20
Project structure Unity realization	Пн 20.01.20	Пн 11.05.20
Furniture models development	Пн 24.02.20	Пн 04.05.20
AR functionality development	Пн 06.04.20	Пт 24.04.20
Testing	**Вт 12.05.20**	**Ср 20.05.20**
Dev Testing	Вт 12.05.20	Чт 14.05.20
QA Testing	Чт 14.05.20	Ср 20.05.20
Ending	**Чт 02.01.20**	**Пт 22.05.20**
Project Documentation	Чт 02.01.20	Ср 20.05.20
Commissioning	Чт 21.05.20	Чт 21.05.20
Archiving the project	Пт 22.05.20	Пт 22.05.20

Рисунок Б.7 – Список робіт діаграми Ганта

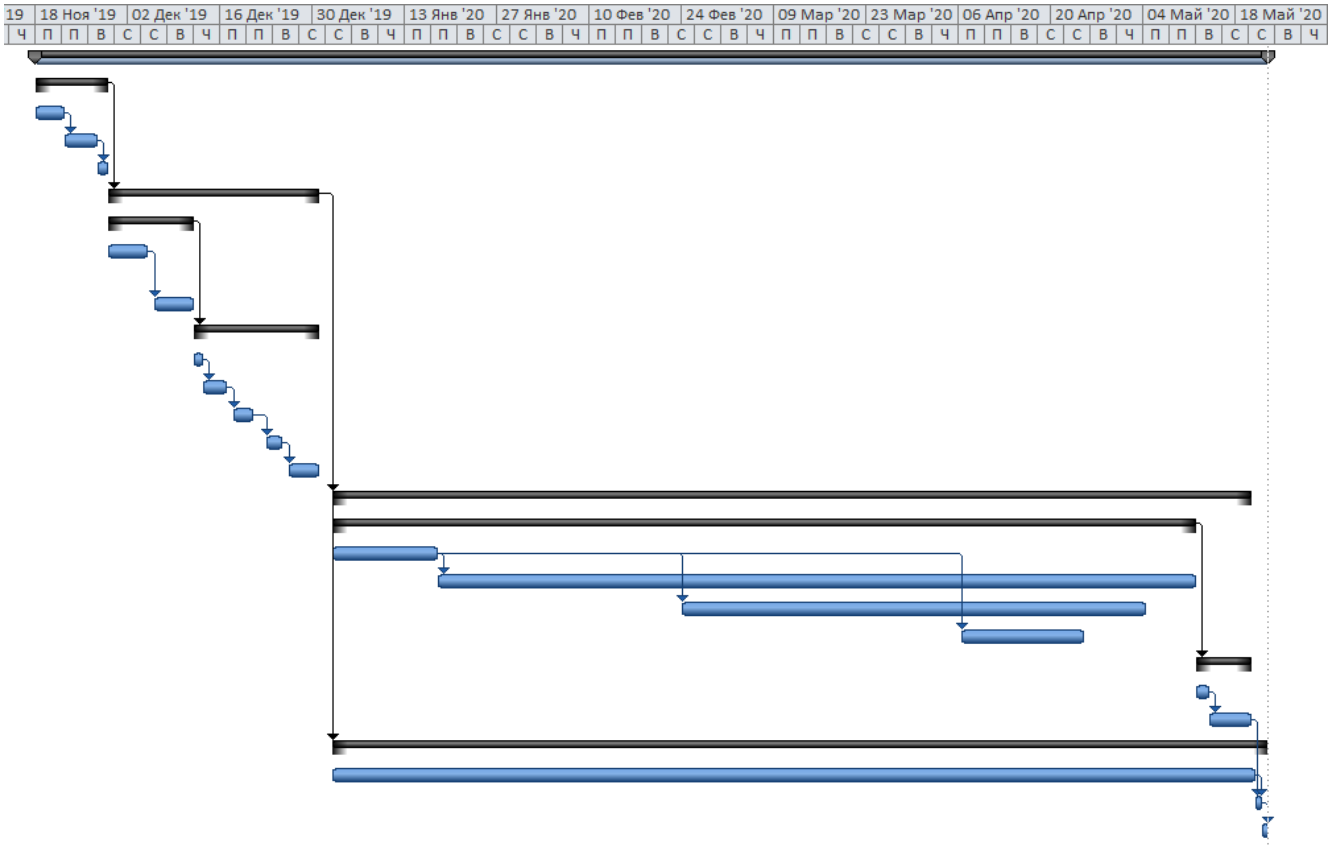

Рисунок Б.8 – Діаграма Ганта

Управління ризиками

Проектний ризик - це ймовірнісна подія або умова, яка може мати позитивний чи негативний вплив на проект. Процеси управління ризиками проекту включають в себе процеси планування управління ризиками, ідентифікації, аналізу, планування реагування та контролю ризиків на проекті.

В процесі ідентифікації було виділено ряд ризиків, що можуть виникнути на проекті. Далі необхідно було провести аналіз ризиків та оцінити ймовірність виникнення та вплив на проект за шкалою від 1 до 5, де 1 – дуже низька оцінка, а 5 – відповідно, максимальна.

На основі проведеного оцінення ймовірності та впливу кожного з визначених ризиків в рамках проекту створюється матриця (рис. Б.10), за якою визначається рівень ризику (Probability / Impact Matrix). Результати проведеної роботи з управління ризиками представлено на рис. Б.9.

Risk ID	Risk description	Probability	Impact	Level of risk	Mitigation
R1	Change of defined requirements	2	4	M	Preparation of a detailed specification. Discuss and approve it with customer
R2	Inability to adhere to the calendar plan	1	3	L	Creating a project implementation plan based on a thorough analysis of all works. Discussing the plan with all team members and approving of the calendar plan with the customer
R3	Occurrence of unplanned work	2	3	M	Approving with customer the list of all work to avoid the appearance of unplanned. Re-planning and optimization of time in case of unplanned works
R4	Software problems	4	4	H	Reinstall the system on time, make updates of required soft, use licensed software, provide backups on external/cloud media
R5	Hardware problems	4	4	H	Make preventive checks, provide project backups on external/cloud media
R6	Misunderstanding in the team	2	2	L	Be able to quickly identify the cause of misunderstandings and eliminate it.
R7	Incorrect prioritization of tasks	4	3	M	Devote time to properly prioritizing tasks. Optimize task priorities

Рисунок Б.9 – Risk Register

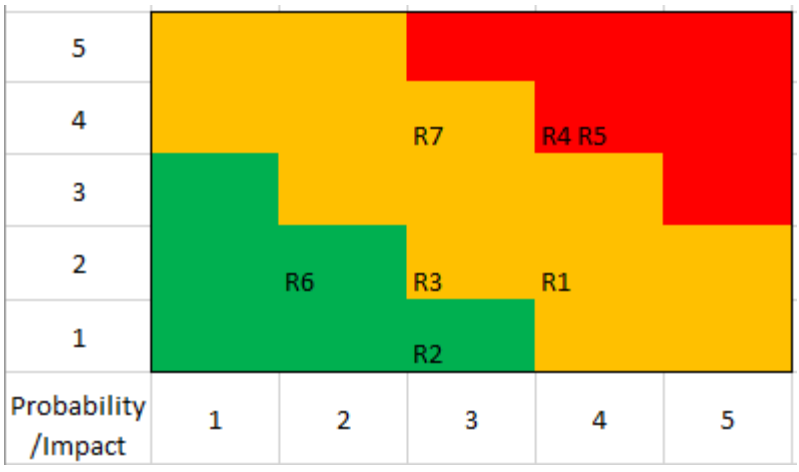

Рисунок Б.10 – Probability Impact Risk Matrix

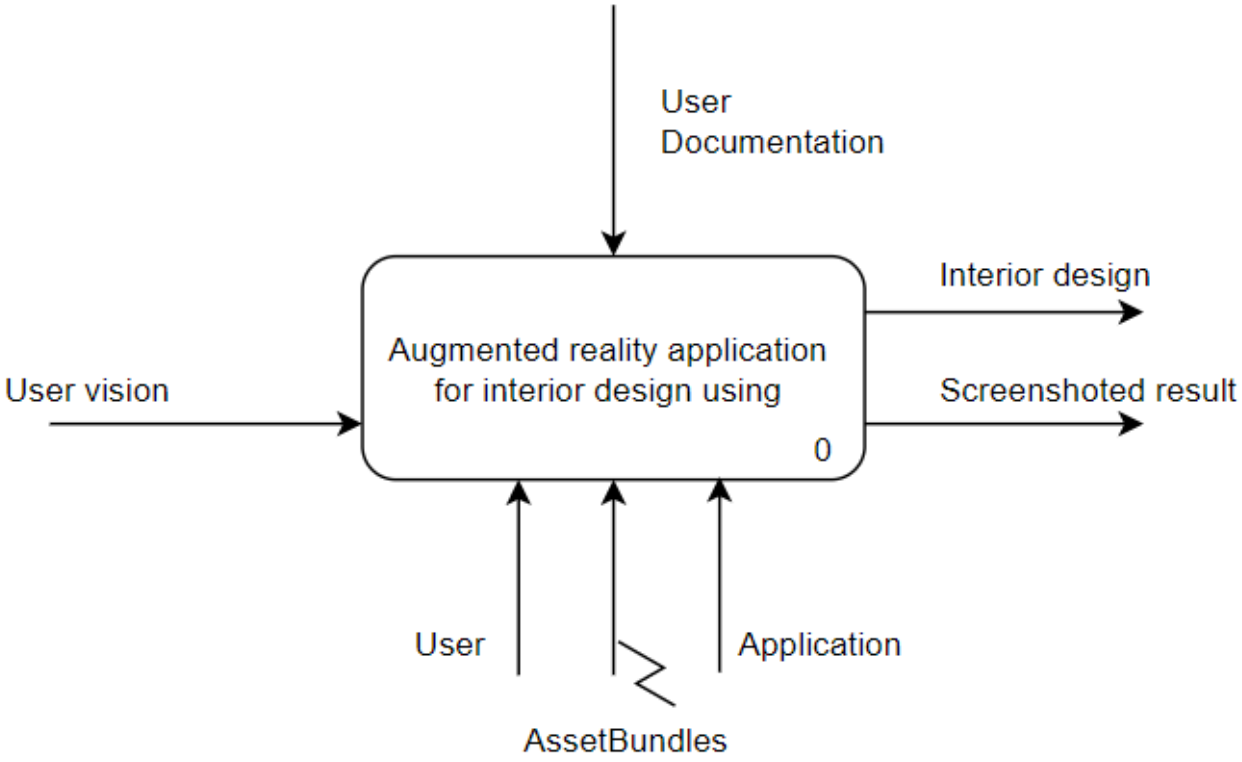

Рисунок Б.11 – Функціональна контекстна діаграма

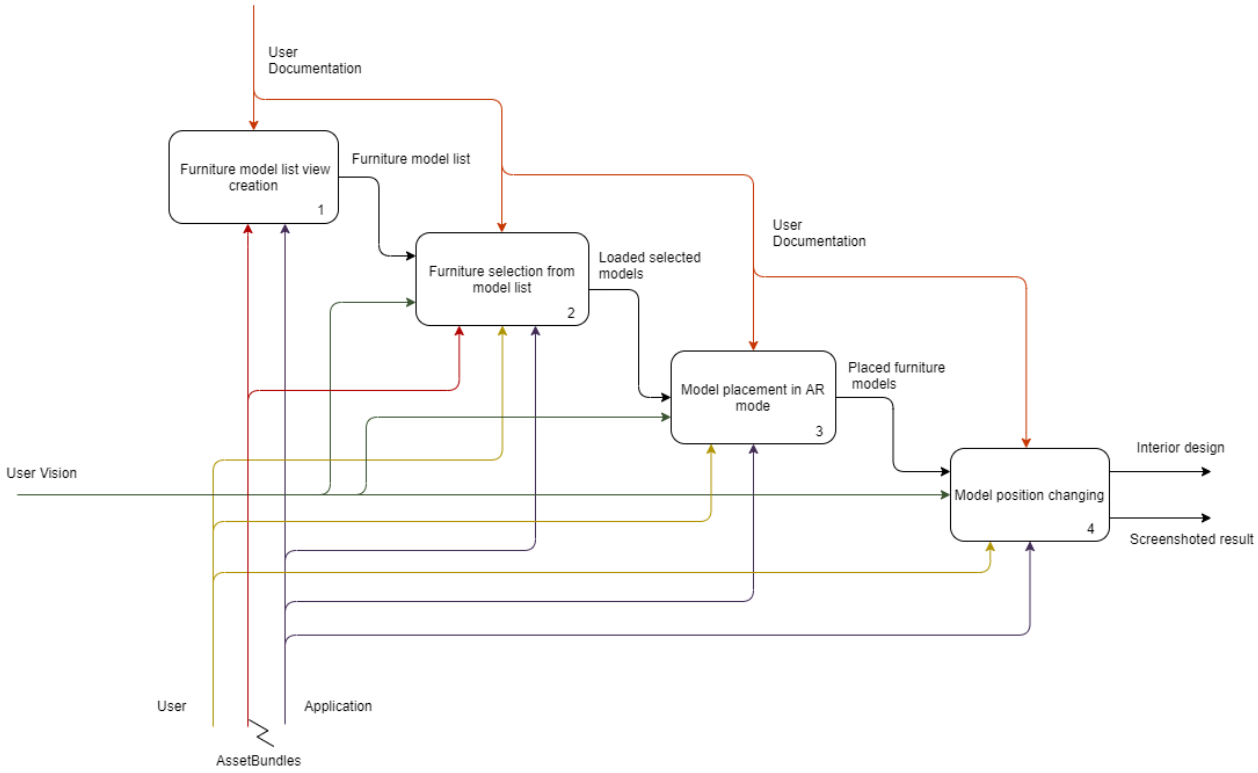

Рисунок Б.12 – Декомпозиція функціональної контекстної діаграми

ДОДАТОК В

Файли коду реалізації

PlaneManager.cs

```
using System.Timers;
using UnityEngine;
using Vuforia;

public class PlaneManager : MonoBehaviour
{
    #region PUBLIC_MEMBERS
    public static bool GroundPlaneHitReceived { get; private set; }
    #endregion // PUBLIC_MEMBERS


    #region PRIVATE_MEMBERS
    [SerializeField] PlaneFinderBehaviour planeFinder = null;

    [Header("Placement Augmentation")]
    [SerializeField] GameObject placementAugmentation = null;

    const string UnsupportedDeviceTitle = "Unsupported Device";
    const string UnsupportedDeviceBody =
        "This device has failed to start the Positional Device Tracker. " +
        "Please check the list of supported devices: " +
        "\n\nhttps://library.vuforia.com/articles/Solution/ground-plane-supported-devices.html";

    StateManager stateManager;
    SmartTerrain smartTerrain;
    PositionalDeviceTracker positionalDeviceTracker;
    ContentPositioningBehaviour contentPositioningBehaviour;
    TouchHandler touchHandler;
    ProductPlacement productPlacement;
    GroundPlaneUI groundPlaneUI;
    AnchorBehaviour placementAnchor;
    int automaticHitTestFrameCount;
    static TrackableBehaviour.Status StatusCached = TrackableBehaviour.Status.NO_POSE;
    static TrackableBehaviour.StatusInfo StatusInfoCached = TrackableBehaviour.StatusInfo.UNKNOWN;

    // More Strict: Property returns true when Status is Tracked and StatusInfo is Normal.
    public static bool TrackingStatusIsTrackedAndNormal
    {
        get
        {
            return
                (StatusCached == TrackableBehaviour.Status.TRACKED ||
                 StatusCached == TrackableBehaviour.Status.EXTENDED_TRACKED) &&
                StatusInfoCached == TrackableBehaviour.StatusInfo.NORMAL;
        }
    }

    // Less Strict: Property returns true when Status is Tracked/Normal or Limited/Unknown.
    public static bool TrackingStatusIsTrackedOrLimited
    {
```

```
        get
        {
            return
                ((StatusCached == TrackableBehaviour.Status.TRACKED ||
                  StatusCached == TrackableBehaviour.Status.EXTENDED_TRACKED) &&
                  StatusInfoCached == TrackableBehaviour.StatusInfo.NORMAL) ||
                 (StatusCached == TrackableBehaviour.Status.LIMITED &&
                  StatusInfoCached == TrackableBehaviour.StatusInfo.UNKNOWN);
        }
    }

    bool SurfaceIndicatorVisibilityConditionsMet
    {
        // The Surface Indicator should only be visible if the following conditions
        // are true:
        // 1. Tracking Status is Tracked or Limited (sufficient for Hit Test Anchors
        // 2. Ground Plane Hit was received for this frame
        // 3. If the Plane Mode is equal to PLACEMENT and *there's no active touches
        get
        {
            return
                (TrackingStatusIsTrackedOrLimited &&
                 GroundPlaneHitReceived &&
                 (Input.touchCount == 0));
        }
    }

    Timer timer;
    bool timerFinished;
    #endregion // PRIVATE_MEMBERS

    #region MONOBEHAVIOUR_METHODS
    void Start()
    {
        VuforiaARController.Instance.RegisterVuforiaStartedCallback(OnVuforiaStarted);
        VuforiaARController.Instance.RegisterOnPauseCallback(OnVuforiaPaused);
        DeviceTrackerARController.Instance.RegisterTrackerStartedCallback(OnTrackerStarted);

DeviceTrackerARController.Instance.RegisterDevicePoseStatusChangedCallback(OnDevicePoseStatusChange
d);

        this.planeFinder.HitTestMode = HitTestMode.AUTOMATIC;
        this.contentPositioningBehaviour = this.planeFinder.GetComponent<ContentPositioningBehaviour>();

        this.productPlacement = FindObjectOfType<ProductPlacement>();
        this.touchHandler = FindObjectOfType<TouchHandler>();
        this.groundPlaneUI = FindObjectOfType<GroundPlaneUI>();
        // Setup a timer to restart the DeviceTracker if tracking does not receive
        // status change from StatusInfo.RELOCALIZATION after 10 seconds.
        this.timer = new Timer(10000);
        this.timer.Elapsed += TimerFinished;
        this.timer.AutoReset = false;
    }

    void Update()
    {
        // The timer runs on a separate thread and we need to ResetTrackers on the main thread.
        if (this.timerFinished)
```

```
        {
            ResetTrackers();
            this.timerFinished = false;
        }
    }

    void LateUpdate()
    {
        // The AutomaticHitTestFrameCount is assigned the Time.frameCount in the
        // HandleAutomaticHitTest() callback method. When the LateUpdate() method
        // is then called later in the same frame, it sets GroundPlaneHitReceived
        // to true if the frame number matches. For any code that needs to check
        // the current frame value of GroundPlaneHitReceived, it should do so
        // in a LateUpdate() method.
        GroundPlaneHitReceived = (this.automaticHitTestFrameCount == Time.frameCount);

        // Surface Indicator visibility conditions rely upon GroundPlaneHitReceived,
        // so we will move this method into LateUpdate() to ensure that it is called
        // after GroundPlaneHitReceived has been updated in Update().
        SetSurfaceIndicatorVisible(SurfaceIndicatorVisibilityConditionsMet);
    }

    void OnDestroy()
    {
        Debug.Log("OnDestroy() called.");

        VuforiaARController.Instance.UnregisterVuforiaStartedCallback(OnVuforiaStarted);
        VuforiaARController.Instance.UnregisterOnPauseCallback(OnVuforiaPaused);
        DeviceTrackerARController.Instance.UnregisterTrackerStartedCallback(OnTrackerStarted);

DeviceTrackerARController.Instance.UnregisterDevicePoseStatusChangedCallback(OnDevicePoseStatusChan
ged);
    }
    #endregion // MONOBEHAVIOUR_METHODS

    #region GROUNDPLANE_CALLBACKS
    public void HandleAutomaticHitTest(HitTestResult result)
    {
        if (productPlacement.furnitureModel == null)
        {
            return;
        }
        this.automaticHitTestFrameCount = Time.frameCount;

        if (!productPlacement.IsPlaced)
        {
            this.productPlacement.SetProductAnchor(null);
            this.placementAugmentation.PositionAt(result.Position);
        }
    }

    public void HandleInteractiveHitTest(HitTestResult result)
    {
        if (productPlacement.furnitureModel == null)
        {
            return;
        }
```

```
    if (result == null)
    {
        Debug.LogError("Invalid hit test result!");
        return;
    }

    if (!groundPlaneUI.IsCanvasButtonPressed())
    {
        Debug.Log("HandleInteractiveHitTest() called.");

        // If the PlaneFinderBehaviour's Mode is Automatic, then the Interactive HitTestResult will be centered.
        this.contentPositioningBehaviour.DuplicateStage = false;

        // With initial tap or a double-tap, a new anchor is created,
        // so we first check that Status=TRACKED/EXTENDED_TRACKED and StatusInfo=NORMAL
        // before proceeding.
        if (TrackingStatusIsTrackedAndNormal)
        {
            if(!this.placementAnchor)
            {
                ResetAugmentation();
            }
            // If the product is yet to be placed when a tap occurs,we assign our stage content, set an
            //anchor and enable the. If a double-tap occurs, for instance when the content has already
            //been placed and positioned, then a new anchor is placed and the stage is centered to it, but
            //the content retains its offset in relation to the stage.
            if (!this.productPlacement.IsPlaced || TouchHandler.DoubleTap)
            {
                this.contentPositioningBehaviour.AnchorStage = this.placementAnchor;
                this.contentPositioningBehaviour.PositionContentAtPlaneAnchor(result);
                UtilityHelper.EnableRendererColliderCanvas(placementAugmentation, true);
            }

            // Immediately following the steps above, we again confirm that the IsPlaced flag has not
            // been set and call SetProductAnchor to reset the transform and rotation of the content in
            // relation to the stage collision plane. This only happens when setting the first anchor or
            // when reseting the scene. When double-tapping to simply create a new anchor, the content
            // retains its positional offset in relation to the stage collision plane as well as its rotation.
            // The SetProductAnchor() will set the IsPlaced flag to true if the transform argument is valid
            // and to false if it is null.
            if (!this.productPlacement.IsPlaced)
            {
                this.productPlacement.SetProductAnchor(this.placementAnchor.transform);
                this.touchHandler.enableRotation = true;
            }
        }
    }
}
#endregion // GROUNDPLANE_CALLBACKS

#region PUBLIC_BUTTON_METHODS
public void SetMode(GameObject furnitureModel)
{
    if (!productPlacement.IsPlaced && productPlacement.furnitureModel != null)
    {
        productPlacement.SetVisible(false);
    }
    touchHandler.augmentationObject = furnitureModel.transform;
```

```
    touchHandler.SetCachedAugmentations();

    productPlacement.furnitureModel = furnitureModel;
    ResetAugmentation();
    productPlacement.OnModelChanged();
}

public void UnsetModel()
{
    productPlacement.furnitureModel = null;
}

/// <summary>
/// This method resets the augmentations and scene elements.
/// It is called by the UI Reset Button and also by OnVuforiaPaused() callback.
/// </summary>
public void ResetScene()
{
    Debug.Log("ResetScene() called.");
    if (this.productPlacement.furnitureModel != null)
    {
        this.productPlacement.Reset();
        UtilityHelper.EnableRendererColliderCanvas(this.placementAugmentation, false);

        this.productPlacement.SetProductAnchor(null);
    }
    this.groundPlaneUI.Reset();
    this.touchHandler.enableRotation = false;
}
#endregion // PUBLIC_BUTTON_METHODS

#region PRIVATE_METHODS
private void ResetAugmentation()
{
    this.placementAugmentation = this.productPlacement.furnitureModel;
    this.placementAnchor = this.placementAugmentation.GetComponentInParent<AnchorBehaviour>();
}

/// <summary>
/// This method can be used to set the Ground Plane surface indicator visibility.
/// This sample will display it when the Status=TRACKED and StatusInfo=Normal.
/// </summary>
/// <param name="isVisible">bool</param>
void SetSurfaceIndicatorVisible(bool isVisible)
{
    Renderer[] renderers = this.planeFinder.PlaneIndicator.GetComponentsInChildren<Renderer>(true);
    Canvas[] canvas = this.planeFinder.PlaneIndicator.GetComponentsInChildren<Canvas>(true);

    foreach (Canvas c in canvas)
        c.enabled = isVisible;

    foreach (Renderer r in renderers)
        r.enabled = isVisible;
}

/// <summary>
/// This is a C# delegate method for the Timer:
/// ElapsedEventHandler(object sender, ElapsedEventArgs e)
```

```
    /// </summary>
    void TimerFinished(System.Object source, ElapsedEventArgs e)
    {
        this.timerFinished = true;
    }
    #endregion // PRIVATE_METHODS
}
```

ProductPlacement.cs

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

public class ProductPlacement : MonoBehaviour
{
    #region PUBLIC_MEMBERS
    public bool IsPlaced { get; private set; }

    [Header("Augmentation Objects")]
    [SerializeField] public GameObject furnitureModel = null;

    [Header("Control Indicators")]
    [SerializeField] public GameObject translationIndicator = null;
    [SerializeField] public GameObject rotationIndicator = null;

    [Header("Augmentation Size")]
    [Range(0.1f, 2.0f)]
    [SerializeField] public float productSize = 1.0f;
    #endregion // PUBLIC_MEMBERS

    #region PRIVATE_MEMBERS
    //For information about Unity render system magic numbers, please visit
https://docs.unity3d.com/Manual/SL-SubShaderTags.html
    public static readonly int OpaqueRenderMode = 0;
    public static readonly int TransperentRenderMode = 3;
    public static readonly int TransperentQueue = 3002;

    private MeshRenderer[] _renderers;
    private List<Material> _materials;

    private GroundPlaneUI _groundPlaneUI;
    private Camera _mainCamera;
    private Ray _cameraToPlaneRay;
    private RaycastHit _cameraToPlaneHit;

    private float _augmentationScale;
    private Vector3 _productScale;
    private string _floorName;

    // Property which returns whether chair visibility conditions are met
    bool VisibilityConditionsMet
    {
        // Model should only be visible if the following conditions are met:
        // 1. Tracking Status is Tracked or Limited
        // 2. Ground Plane Hit was received for this frame
        get
```

```
        {
            return
                PlaneManager.TrackingStatusIsTrackedOrLimited &&
                PlaneManager.GroundPlaneHitReceived;
        }
    }
    #endregion // PRIVATE_MEMBERS

    #region MONOBEHAVIOUR_METHODS
    void Start()
    {
        this._mainCamera = Camera.main;
        this._groundPlaneUI = FindObjectOfType<GroundPlaneUI>();
        this._augmentationScale = VuforiaRuntimeUtilities.IsPlayMode() ? 0.1f : this.productSize;
    }

    void Update()
    {
        if (this.furnitureModel == null)
        {
            return;
        }

        EnablePreviewModeTransparency(!this.IsPlaced);
        if (!this.IsPlaced)
            UtilityHelper.RotateTowardCamera(this.furnitureModel);

        if (this.IsPlaced)
        {
            this.rotationIndicator.SetActive(Input.touchCount == 2);

            this.translationIndicator.SetActive(
                (TouchHandler.IsSingleFingerDragging || TouchHandler.IsSingleFingerStationary) &&
!this._groundPlaneUI.IsCanvasButtonPressed());

            if (TouchHandler.IsSingleFingerDragging || (VuforiaRuntimeUtilities.IsPlayMode() &&
Input.GetMouseButton(0)))
            {
                if (!this._groundPlaneUI.IsCanvasButtonPressed())
                {
                    this._cameraToPlaneRay = this._mainCamera.ScreenPointToRay(Input.mousePosition);

                    if (Physics.Raycast(this._cameraToPlaneRay, out this._cameraToPlaneHit))
                    {
                        if (this._cameraToPlaneHit.collider.gameObject.name == _floorName)
                        {
                            this.furnitureModel.PositionAt(this._cameraToPlaneHit.point);
                        }
                    }
                }
            }
        }
        else
        {
            this.rotationIndicator.SetActive(false);
            this.translationIndicator.SetActive(false);
        }
    }
```

```
void LateUpdate()
{
    if (this.furnitureModel == null)
    {
        return;
    }

    if (!this.IsPlaced)
    {
        SetVisible(this.VisibilityConditionsMet);
    }
}
#endregion // MONOBEHAVIOUR_METHODS

#region PUBLIC_METHODS
public void Reset()
{
    this.furnitureModel.transform.position = Vector3.zero;
    this._productScale =
        new Vector3(this._augmentationScale * this.furnitureModel.transform.localScale.x,
                this._augmentationScale * this.furnitureModel.transform.localScale.y,
                this._augmentationScale * this.furnitureModel.transform.localScale.z);
    this.furnitureModel.transform.localScale = this._productScale;
}

public void SetProductAnchor(Transform transform)
{
    if (transform)
    {
        this.IsPlaced = true;
        this.furnitureModel.transform.SetParent(transform);
        this.furnitureModel.transform.localPosition = Vector3.zero;
        UtilityHelper.RotateTowardCamera(this.furnitureModel);
    }
    else
    {
        this.IsPlaced = false;
    }
}

public void OnModelChanged()
{
    ResetRenderers();
    if (!this._renderers[0].enabled)
    {
        Reset();
        this.IsPlaced = this._renderers[0].enabled;
    }

    SetupFloor();
    SetControllIndicators();
}

public void SetControllIndicators()
{
        this.rotationIndicator.transform.SetParent(this.furnitureModel.transform);
        this.rotationIndicator.transform.localPosition = Vector3.zero;
```

```
        this.rotationIndicator.transform.localEulerAngles = new Vector3(90.0f, 0.0f, 0.0f);

        this.translationIndicator.transform.SetParent(this.furnitureModel.transform);
        this.translationIndicator.transform.localPosition = Vector3.zero;
        this.translationIndicator.transform.localEulerAngles = new Vector3(90.0f, 0.0f, 0.0f);
    }
    #endregion // PUBLIC_METHODS

    #region PRIVATE_METHODS
    private void ResetRenderers()
    {
        this._renderers = this.furnitureModel.GetComponentsInChildren<MeshRenderer>();
        QueryMaterials();
        ChangeModelRenderModel();
    }

    void SetupFloor()
    {
        if (VuforiaRuntimeUtilities.IsPlayMode())
        {
            this._floorName = "Emulator Ground Plane";
        }
        else
        {
            if (this.furnitureModel == null)
            {
                return;
            }
            this._floorName = "Floor";
            GameObject floor = GetFloor();
            floor.transform.SetParent(this.furnitureModel.transform.parent);
            floor.transform.SetPositionAndRotation(Vector3.zero, Quaternion.identity);
            floor.transform.localScale = Vector3.one;
            floor.GetComponent<BoxCollider>().size = new Vector3(100f, 0, 100f);
        }
    }

    GameObject GetFloor()
    {
        return FindObjectOfType<BoxCollider>()?.gameObject ?? new GameObject(this._floorName,
typeof(BoxCollider));
    }

    /// <summary>
    /// This method is used prior to model being placed. Once placed, model visibility is controlled
    /// by the DefaultTrackableEventHandler.
    /// </summary>
    public void SetVisible(bool visible)
    {
        // Set the visibility of the model
        foreach (Renderer r in _renderers)
            r.enabled = visible;
    }

    private void QueryMaterials()
    {
        _materials = new List<Material>();
        foreach (Transform subModel in furnitureModel.transform)
```

```
        {
            if(subModel.GetComponent<Renderer>() == null)
            {
                continue;
            }
            foreach (Material material in subModel.GetComponent<Renderer>().materials)
            {
                _materials.Add(material);
            }
        }
    }
    private void ChangeModelRenderModel()
    {
        foreach (var material in _materials)
        {
            material.SetFloat("_Mode", TransperentRenderMode);
            material.SetInt("_SrcBlend", (int)UnityEngine.Rendering.BlendMode.SrcAlpha);
            material.SetInt("_DstBlend", (int)UnityEngine.Rendering.BlendMode.OneMinusSrcAlpha);

            material.DisableKeyword("_ALPHATEST_ON");
            material.DisableKeyword("_ALPHABLEND_ON");
            material.EnableKeyword("_ALPHAPREMULTIPLY_ON");

            material.renderQueue = TransperentQueue;
        }
    }

    void EnablePreviewModeTransparency(bool previewEnabled)
    {
        foreach (var m in _materials)
        {
            m.SetFloat("_Mode", previewEnabled ? TransperentRenderMode : OpaqueRenderMode);
            m.color = previewEnabled ? new Color(m.color.r, m.color.g, m.color.b, 0.4f) : new Color(m.color.r,
m.color.g, m.color.b, 1.0f);
        }
    }
    #endregion // PRIVATE_METHODS

}
```

GroundPlaneUI.cs

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

public class GroundPlaneUI : MonoBehaviour
{
    [Header("UI Elements")]
    [SerializeField] public Text title = null;
    [SerializeField] public Text instructions = null;
    [SerializeField] public CanvasGroup screenReticle = null;

    [SerializeField] public Toggle togglePlacementPrefab;
```

```
    private bool resetDefaultToggle = true;
    private const string TitleProductPlacement = "InteriAR";

    private ToggleGroup _toggleGroup;
    private GraphicRaycaster _graphicRayCaster;
    private PointerEventData _pointerEventData;
    private EventSystem _eventSystem;

    private Canvas _commonUi;
    private GameObject _mainCanvas;
    private GameObject _menuCanvas;

    private ProductPlacement _productPlacement;
    private TouchHandler _touchHandler;
    private FirebaseApp app;

    #region MONOBEHAVIOUR_METHODS
    void Awake()
    {
        _toggleGroup = this.GetComponentInChildren<ToggleGroup>();
        _commonUi = title.GetComponentInParent<Canvas>();
        _mainCanvas = GameObject.Find("MainCanvas");
        _menuCanvas = GameObject.Find("MenuCanvas");
    }

    void Start()
    {
        this.title.text = TitleProductPlacement;

        this._productPlacement = FindObjectOfType<ProductPlacement>();
        this._touchHandler = FindObjectOfType<TouchHandler>();
        this._graphicRayCaster = FindObjectOfType<GraphicRaycaster>();
        this._eventSystem = FindObjectOfType<EventSystem>();

DeviceTrackerARController.Instance.RegisterDevicePoseStatusChangedCallback(OnDevicePoseStatusChange
d);
    }

    void LateUpdate()
    {
        if (PlaneManager.GroundPlaneHitReceived && PlaneManager.TrackingStatusIsTrackedAndNormal)
        {
            // We got an automatic hit test this frame
            // Hide the onscreen reticle when we get a hit test
            this.screenReticle.alpha = 0;

            this.instructions.transform.parent.gameObject.SetActive(true);
            this.instructions.enabled = true;

            this.instructions.text = (_toggleGroup.transform.childCount == 0) ? "Press + to select furniture to place"
:
                (this._productPlacement.IsPlaced) ?
                "• Touch and drag to move" +
                "\n• Two fingers to rotate" +
                ((this._touchHandler.enablePinchScaling) ? " or pinch to scale" : "") +
                "\n• Double-tap to reset Anchor location"
                :
```

```
            "Tap to place";
        }
        else
        {
            // No automatic hit test, so set alpha based on which plane mode is active
            if (!PlaneManager.GroundPlaneHitReceived)
            {
                this.screenReticle.alpha = 1;
            }

            this.instructions.transform.parent.gameObject.SetActive(true);
            this.instructions.enabled = true;

            this.instructions.text = (_toggleGroup.transform.childCount == 0) ? "Press + to select furniture to place"
:
                PlaneManager.GroundPlaneHitReceived ?
                "Move to get better tracking for placing an anchor" :
                "Point device towards ground";
        }
    }

    void OnDestroy()
    {
        Debug.Log("OnDestroy() called.");

DeviceTrackerARController.Instance.UnregisterDevicePoseStatusChangedCallback(OnDevicePoseStatusChan
ged);
    }
    #endregion // MONOBEHAVIOUR_METHODS

    #region PUBLIC_METHODS
    public Toggle AddToggleItem()
    {
        var instance = Instantiate(togglePlacementPrefab, _toggleGroup.transform, false);
        instance.group = _toggleGroup;
        return instance;
    }

    public void Reset()
    {
        this.resetDefaultToggle = true;
    }

    public void UpdateTitle()
    {
        this.title.text = TitleProductPlacement;
    }

    public bool IsCanvasButtonPressed()
    {
        _pointerEventData = new PointerEventData(this._eventSystem)
        {
            position = Input.mousePosition
        };
        List<RaycastResult> results = new List<RaycastResult>();
        this._graphicRayCaster.Raycast(_pointerEventData, results);

        bool resultIsButton = false;
```

```
        foreach (RaycastResult result in results)
        {
            if (result.gameObject.GetComponentInParent<Toggle>() ||
                result.gameObject.GetComponent<Button>())
            {
                resultIsButton = true;
                break;
            }
        }
        return resultIsButton;
    }

    public void CameraButtonHandle()
    {
        StartCoroutine(AndroidShareToolCaller.ShareScreenshot());
    }

    public void AddButtonHandler(bool isOn)
    {
        _commonUi.gameObject.SetActive(!isOn);
        _touchHandler.gameObject.SetActive(!isOn);
        _mainCanvas.gameObject.SetActive(!isOn);
        _menuCanvas.gameObject.SetActive(isOn);
    }
    #endregion // PUBLIC_METHODS
}
```

TouchHandler.cs

```
using UnityEngine;

public class TouchHandler : MonoBehaviour
{
    #region PUBLIC_MEMBERS
    public Transform augmentationObject = null;

    [HideInInspector]
    public bool enableRotation;
    public bool enablePinchScaling;

    public static bool DoubleTap
    {
        get { return (Input.touchSupported) && Input.touches[0].tapCount == 2; }
    }

    public static bool IsSingleFingerStationary
    {
        get { return IsSingleFingerDown() && (Input.touches[0].phase == TouchPhase.Stationary); }
    }

    public static bool IsSingleFingerDragging
    {
        get { return IsSingleFingerDown() && (Input.touches[0].phase == TouchPhase.Moved); }
    }

    const float ScaleRangeMin = 0.1f;
    const float ScaleRangeMax = 2.0f;
```

```
    Touch[] touches;
    static int lastTouchCount;
    bool isFirstFrameWithTwoTouches;
    float cachedTouchAngle;
    float cachedTouchDistance;
    float cachedAugmentationScale;
    Vector3 cachedAugmentationRotation;

    public void SetCachedAugmentations()
    {
        this.cachedAugmentationScale = this.augmentationObject.localScale.x;
        this.cachedAugmentationRotation = this.augmentationObject.localEulerAngles;
    }

    void Update()
    {
        if(augmentationObject == null)
        {
            return;
        }

        this.touches = Input.touches;

        if (Input.touchCount == 2)
        {
            float currentTouchDistance = Vector2.Distance(this.touches[0].position, this.touches[1].position);
            float diff_y = this.touches[0].position.y - this.touches[1].position.y;
            float diff_x = this.touches[0].position.x - this.touches[1].position.x;
            float currentTouchAngle = Mathf.Atan2(diff_y, diff_x) * Mathf.Rad2Deg;

            if (this.isFirstFrameWithTwoTouches)
            {
                this.cachedTouchDistance = currentTouchDistance;
                this.cachedTouchAngle = currentTouchAngle;
                this.isFirstFrameWithTwoTouches = false;
            }

            float angleDelta = currentTouchAngle - this.cachedTouchAngle;
            float scaleMultiplier = (currentTouchDistance / this.cachedTouchDistance);
            float scaleAmount = this.cachedAugmentationScale * scaleMultiplier;
            float scaleAmountClamped = Mathf.Clamp(scaleAmount, ScaleRangeMin, ScaleRangeMax);

            if (this.enableRotation)
            {
                this.augmentationObject.localEulerAngles = this.cachedAugmentationRotation - new Vector3(0,
angleDelta * 3f, 0);
            }
            if (this.enableRotation && this.enablePinchScaling)
            {
                // Optional Pinch Scaling can be enabled via Inspector for this Script Component
                this.augmentationObject.localScale = new Vector3(scaleAmountClamped, scaleAmountClamped,
scaleAmountClamped);
            }

        }
        else if (Input.touchCount < 2)
        {
```

```
            SetCachedAugmentations();
            this.isFirstFrameWithTwoTouches = true;
        }
        else if (Input.touchCount == 6)
        {
            // enable runtime testing of pinch scaling
            this.enablePinchScaling = true;
        }
        else if (Input.touchCount == 5)
        {
            // disable runtime testing of pinch scaling
            this.enablePinchScaling = false;
        }
    }

    static bool IsSingleFingerDown()
    {
        if (Input.touchCount == 0 || Input.touchCount >= 2)
            lastTouchCount = Input.touchCount;

        return (
            Input.touchCount == 1 &&
            Input.touches[0].fingerId == 0 &&
            lastTouchCount == 0);
    }
}
```

ListView.cs

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Collections;
using System;
using Firebase.Storage;


public class ListView : MonoBehaviour
{
    [SerializeField]
    public VerticalLayoutGroup categoryContainerPrefab;
    [SerializeField]
    public Button itemButtonPrefab;

    private GameObject _content;
    private RuntimeLoader _runtimeLoader;
    private GameObject _progressBar;

    private Dictionary<(string, Hash128), List<string>> _categoriesNames = new Dictionary<(string, Hash128),
List<string>>();
    private string _cloudAssetBundles = "gs://interiar-42388.appspot.com/AssetBundles/";
    private string _bundleUrl;

    private FirebaseStorage _firebaseStorage;
```

```
    void Awake()
    {
        _content = this.transform.Find("Viewport/Content").gameObject;
        _runtimeLoader = FindObjectOfType<RuntimeLoader>();
        _progressBar = GameObject.Find("ProgressBarCanvas");
    }


    // Use this for initialization
    IEnumerator Start()
    {
        _progressBar.SetActive(true);

        // Get a reference to the storage service, using the default Firebase App
        _firebaseStorage = FirebaseStorage.DefaultInstance;

        StorageReference manifestRef =
_firebaseStorage.GetReferenceFromUrl($"{_cloudAssetBundles}AssetBundles.manifest");
        var task = manifestRef.GetDownloadUrlAsync();
        string manifestUri = default;
        yield return new WaitUntil(() => task.IsCompleted);
        if (!task.IsFaulted && !task.IsCanceled)
        {
            Debug.Log("Download URL: " + task.Result);
            // ... now download the file via WWW or UnityWebRequest.
            manifestUri = task.Result.AbsoluteUri;
        }
        WWW wWW = new WWW(manifestUri);
        yield return wWW;
        var hashRow = wWW?.text.ToString().Split("\n".ToCharArray())[5];
        var hash = Hash128.Parse(hashRow.Split(':')[1].Trim());

        if (hash.isValid == true)
        {
            //Create a reference from a Google Cloud Storage URI
            StorageReference reference =
_firebaseStorage.GetReferenceFromUrl($"{_cloudAssetBundles}AssetBundles");
            // Fetch the download URL

            task = reference.GetDownloadUrlAsync();
            yield return new WaitUntil(() => task.IsCompleted);
            if (!task.IsFaulted && !task.IsCanceled)
            {
                Debug.Log("Download URL: " + task.Result);
                // ... now download the file via WWW or UnityWebRequest.
                yield return StartCoroutine(LoadCategoriesFromAssetBundle(task.Result, hash));
            }
        }
        InitCategories();

        _progressBar.SetActive(false);
    }

    void OnDisable()
    {
        foreach (Transform child in _content.transform)
        {
            child.GetComponentInChildren<Toggle>().isOn = false;
```

```
        }
    }

    IEnumerator LoadCategoriesFromAssetBundle(Uri uri, Hash128 hash)
    {
        using (WWW www = WWW.LoadFromCacheOrDownload(uri.AbsoluteUri, hash))
        {
            yield return www;
            if (!string.IsNullOrEmpty(www.error))
            {
                Debug.Log(www.error);
                yield break;
            }
            AssetBundle assetBundle = www.assetBundle;
            AssetBundleManifest manifest =
assetBundle.LoadAsset<AssetBundleManifest>("AssetBundleManifest");
            List<string> categories = manifest.GetAllAssetBundles().ToList();
            Dictionary<string, Hash128> categoriesHash = new Dictionary<string, Hash128>();
            foreach(string category in categories)
            {
                categoriesHash.Add(category, manifest.GetAssetBundleHash(category));
            }
            yield return StartCoroutine(LoadBundles(categoriesHash));
        }
    }

    private IEnumerator LoadBundles(Dictionary<string, Hash128> categories)
    {
        foreach(KeyValuePair<string, Hash128> category in categories)
        {
            _bundleUrl = $"{_cloudAssetBundles}{category.Key}.manifest";
            //Create a reference from a Google Cloud Storage URI
            StorageReference reference = _firebaseStorage.GetReferenceFromUrl(_bundleUrl);
            // Fetch the download URL
            Uri downloadUri = null;
            var task = reference.GetDownloadUrlAsync();
            yield return new WaitUntil(() => task.IsCompleted);
            if (!task.IsFaulted && !task.IsCanceled)
            {
                Debug.Log("Download URL: " + task.Result);
                // ... now download the file via WWW or UnityWebRequest.
                downloadUri = task.Result;
            }

            using (WWW www = new WWW(downloadUri.AbsoluteUri))
            {
                yield return www;
                if (!string.IsNullOrEmpty(www.error))
                {
                    Debug.Log(www.error);
                    yield break;
                }
                string splitString = "Assets:";
                int splitIndex = www.text.IndexOf("Assets:") + splitString.Length;
                string assetsString = www.text.Substring(splitIndex, www.text.IndexOf("Dependencies") -
splitIndex);
                List<string> assetNames = assetsString.Split(new string[] { "\n- ", "\n" },
StringSplitOptions.RemoveEmptyEntries).ToList();
```

```
            _categoriesNames.Add((category.Key, category.Value), assetNames);
        }
    }
}

private void InitCategories()
{
    foreach (KeyValuePair<(string, Hash128), List<string>> category in _categoriesNames)
    {
        var instance = Instantiate(categoryContainerPrefab, _content.transform, false);
        string categoryName = Path.GetFileName(category.Key.Item1);
        categoryName = categoryName.First().ToString().ToUpper() + categoryName.Substring(1);
        CategoryItem categoryItem = new CategoryItem(instance.transform, categoryName);
        foreach (string item in category.Value)
        {
            var button = Instantiate(itemButtonPrefab, categoryItem.LayoutGroup.transform, false);
            string name = Path.GetFileNameWithoutExtension(item);
            string itemTitle = name.First().ToString().ToUpper() + name.Substring(1);
            button.GetComponentInChildren<Text>().text = itemTitle;
            button.onClick.AddListener(() => StartCoroutine(SelectItemEventHandler(category.Key, name)));
        }
    }
}

private IEnumerator SelectItemEventHandler((string, Hash128) category, string modelName)
{
    _progressBar.SetActive(true);
    _bundleUrl = $"{_cloudAssetBundles}{category.Item1}";
    //Create a reference from a Google Cloud Storage URI
    StorageReference reference = _firebaseStorage.GetReferenceFromUrl(_bundleUrl);
    // Fetch the download URL
    var task = reference.GetDownloadUrlAsync();
    yield return new WaitUntil(() => task.IsCompleted);
    if (!task.IsFaulted && !task.IsCanceled)
    {
        Debug.Log("Download URL: " + task.Result);
        // ... now download the file via WWW or UnityWebRequest.
        StartCoroutine(_runtimeLoader.LoadModel(task.Result.AbsoluteUri, category.Item2, modelName));
    }
}

#region Nested class
class CategoryItem
{
    public Toggle CategoryToggle { set; get; }
    public Text Label { set; get; }
    public LayoutGroup LayoutGroup { set; get; }

    public CategoryItem(Transform instance, string labelText)
    {
        Label = instance.GetComponentInChildren<Text>();
        CategoryToggle = instance.GetComponentInChildren<Toggle>();
        LayoutGroup = instance.Find("LayoutGroup").GetComponent<LayoutGroup>();

        Label.text = labelText;
        CategoryToggle.group = instance.GetComponentInParent<ToggleGroup>();
    }
```

```
    }
    #endregion
}
```

RuntimeLoader.cs

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

public class RuntimeLoader : MonoBehaviour
{
    [SerializeField] public GameObject anchorPlacementPrefab;

    private GameObject _anchors;
    private GroundPlaneUI _groundPlaneUI;
    private PlaneManager _planeManager;
    private GameObject _progressBar;
    private AssetBundle _assetBundle;

    void Awake()
    {
        _anchors = GameObject.Find("Anchors");
        _groundPlaneUI = FindObjectOfType<GroundPlaneUI>();
        _planeManager = FindObjectOfType<PlaneManager>();
        _progressBar = GameObject.Find("ProgressBarCanvas");
        _progressBar.gameObject.SetActive(false);
        gameObject.GetComponentInParent<Canvas>().gameObject.SetActive(false);
    }

    public IEnumerator LoadModel(string bundleUrl, Hash128 hash, string modelName)
    {
        while (!Caching.ready)
        {
            yield return null;
        }
        using (WWW www = WWW.LoadFromCacheOrDownload(bundleUrl, hash))
        {
            yield return www;

            if (!string.IsNullOrEmpty(www.error))
            {
                Debug.Log(www.error);
                yield break;
            }

            _assetBundle = www.assetBundle;
            AssetBundleRequest loadRequest = _assetBundle.LoadAssetAsync<GameObject>(modelName);
            yield return loadRequest;

            GameObject parentAnchor = Instantiate(anchorPlacementPrefab, _anchors.transform);
            GameObject furnitureModel = Instantiate(loadRequest.asset as GameObject);

            furnitureModel.transform.SetParent(parentAnchor.transform);
            furnitureModel.transform.localPosition = new Vector3(0.0f, 0.0f, 0.0f);
```

```
            var instance = _groundPlaneUI.AddToggleItem();
            instance.onValueChanged.AddListener(delegate {
                if (instance.isOn)
                {
                    _planeManager.SetMode(furnitureModel);
                }
            });
            instance.transform.Find("Background/Remove").GetComponent<Button>().onClick.AddListener(() =>
            {
                Destroy(instance.gameObject);
                Destroy(parentAnchor.gameObject);
                _planeManager.UnsetModel();
            });
            string itemTitle = modelName.First().ToString().ToUpper() + modelName.Substring(1);
            instance.transform.Find("Text").GetComponent<Text>().text = itemTitle;
        }

        if (_assetBundle != null)
        {
            _assetBundle.Unload(false);
        }

        Debug.Log($"Complete loading {modelName}");
        _progressBar.SetActive(false);
    }
}
```

ДОДАТОК Г

Матеріали конференції

Technologies of augmented reality at the service of interior design of premises

Onoshko V.V., *Student*; Kuznetsov E.G., *Senior Lecturer*
Sumy State University, Sumy, Ukraine

Augmented reality technologies have recently become increasingly part of our daily lives in order to make it convenient and more interesting. Virtual elements can be presented in both 2D and 3D format, depending on user needs.

Today, when 3D modeling technologies are quite developed, we want to get increasingly realistic images of objects. Photogrammetry enables you to create quality 3D photo scans and models, as well as textures.

Dealing with the problem of setting up a room with new furniture, it becomes obvious that a modern person would not be prevented from seeing in advance what one or another thing in the interior will look like before its acquisition.

Today, when augmented reality technologies are rapidly evolving, we can already say that it is possible to change the perception of existing things without any real change. These technologies are best suited to the interior planning process.

The idea of the project is to turn through photogrammetry the sets of photos of real furniture into their 3D models and through augmented reality tools to allow these derived models to be placed in the interior. Based on the idea, the purpose of this work was formed, which is to develop a mobile application of augmented reality for the development of interiors with furniture.

After reviewing the best known gaming processors, a Unity3D was selected to develop the project product. The tools for developing AR applications with Unity were selected the Vuforia AR. Autodesk Recap tools and capabilities were used to create models.

The practical significance of this work is that in its application for interior planning, it will help users without forcing them to buy furniture that may not be suitable enough for the interior being created.

This development can be useful both for regular furniture stores and online stores, can increase sales, reduce the number of returns because customers will be able to make a right choice in advance.

Рисунок Г.1 – Тези зі збірника конференції IMA 2020