Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Яковлєва Владислава Олексіївна*
(ПІБ)

академічної групи *121-17-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка програмного забезпечення під керуванням ОС Android для виміру фізичних розмірів об'єктів за допомогою камери мобільного телефону*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц.Приходченко С.Д.*			
розділів:				
спеціальний	*доц. Приходченко С.Д.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2021

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2021 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-17-1* (група) *Яковлєвій В.О.* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка програмного забезпечення під керуванням ОС Android для виміру фізичних розмірів об'єктівза допомогою камери мобільного телефону*

затверджена наказом ректора НТУ «ДП» від *07.06.2021 р.* № *317-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2021 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2021 р.***

Завдання видав _____ (підпис) *доц. Приходченко С.Д.* (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ (підпис) *Яковлєва В.О.* (прізвище, ініціали)

Дата видачі завдання: *14.01.2021 р.*

Термін подання кваліфікаційної роботи до ЕК: *11.06.2021 р.*

РЕФЕРАТ

Пояснювальна записка: ___ с., ___ рис., ___ табл., __ дод., ___ джерел.

Об'єкт розробки: мобільний додаток на Android, що вимірює фізичні розміри об’єктів за допомогою камери телефона.

Мета кваліфікаційної роботи: створення мобільного додатку на Android-платформі, що надаватиму користувачам змогу вимірювати фізичні реальні об’єкти за допомогою камери телефону на основі технології доповненої реальності.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточняється постановка завдання.

У першому розділі проаналізовано предметну галузь, визначено актуальність завдання та призначення розробки, сформульовано постановку завдання, зазначено вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі проаналізовані наявні рішення, обрано платформи для розробки, виконано проектування і розробка програми, описана робота програми, алгоритм і структура її функціонування, а також виклик та завантаження програми, визначено вхідні і вихідні дані, охарактеризовано склад параметрів технічних засобів.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення полягає у створенні мобільного додатка, що дозволяє вимірювати з точністю реальні об’єкти, зберігати створені заміри.

Актуальність даного програмного продукту визначається великим попитом на подібні розробки, що оптимізують та спрощують дії щодо вимірювання у компаніях для маркетингового відділу або в особистих цілях окремого користувача.

Список ключових слів: МОБІЛЬНИЙ, ДОДАТОК, ANDROID, AR, ДОПОВНЕНА РЕАЛЬНІСТЬ, ТЕЛЕФОН, АЛГОРИТМ, ПРОЕКТУВАННЯ, МЕНЮ, ГАЛЕРЕЯ, КАМЕРА.

ABSTRACT

Explanatory note: ___ p., ___ figs., ___ tabl, __ appx., ___ sources.

Object of development: an Android mobile application that measures the physical size of objects using a phone camera.

Purpose of the qualification work is to create a mobile application on the Android platform, which will allow users to measure physical real objects with the help of a phone camera based on augmented reality technology.

In the introduction it is considers the analysis and the current state of the problem, specifies the purpose of the qualification work and the scope of its application, provides a justification for the relevance of the topic and clarifies the problem.

In the first section the subject branch is analyzed, the urgency of the task and purpose of development are defined, the statement of the task is formulated, requirements to software realization, technologies and software are specified.

In the second section it is analyzes the existing solutions, selected platforms for development, designed and developed the program, describes the program, algorithm and structure of its operation, as well as calling and loading the program, determines the input and output data, describes the parameters of hardware.

In the economic section it is determines the complexity of the developed information system, calculates the cost of work to create a program and calculates the time for its creation.

The practical value is to create a mobile application that allows you to accurately measure real objects, save the created measurements.

The relevance of this software product is determined by the high demand for such developments that optimize and simplify measurement actions in companies for the marketing department or for the personal purposes of an individual user.

List of keywords: MOBILE, APPLICATION, ANDROID, AR, AUGMENTED REALITY, PHONE, ALGORITHM, DESIGN, MENU, GALLERY, CAMERA.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

3D – тривімірний простір (англ. three-dimensional);

APK – Android Package;

AR – доповнена реальність (англ. augmented reality);

MVC – модель-вид-контролер (англ. model-view-controller);

SDK – Software Development Kit;

SLAM – simulations localization and mapping;

VR – віртуальна реальність (англ. virtual reality);

ЕОМ – електронно-обчислювальна машина;

ПЗ – програмне забезпечення.

ВСТУП

Темою даної кваліфікаційної роботи є розробка мобільного додатка з використанням технології доповненої реальності для вимірювання розмірів фізичних об'єктів за допомогою камери телефону.

Метою даної кваліфікаційної роботи є вивчення інструментальних засоби Android Studio, вивчення і визначення різних методів побудова доповненої реальності на базі платформи Android, дослідження кращого способу розробки програми, а також вивчення різних алгоритмів за визначенням об'єктів.

Доповнена реальність як термін був запропонований ще в 1990 році, а розробка цієї технології була розпочата ще раніше. Однак довгий час технологія доповненої реальності стикалася з труднощами в поліпшенні, сумісності зі старими пристроями, математичними численнями, через що процес розвитку зайняв більше передбачуваної кількості часу.

Але вже на початку 2020 року доповнена реальність різко довела свою необхідність. Практично всі сфери стали застосовувати цю технологію в своїх компаніях: від медицини до додатків з дитячими іграми. І це далеко не межа, адже основний потенціал даної технології ще далеко не розкритий.

Інтерес до доповненої реальності зростає з кожним роком. Більшість компаній вже впровадили подібну технологію в свої додатки, що набагато полегшує роботу з покупцями або з об'єктами. Що стосується теми кваліфікаційної роботи, то не можна сказати, що подібних мобільних розробок не існувало раніше. Однак всі вони будувалися на базових алгоритмах, іноді розробникам доводилося винаходити «велосипед», чому кінцевий результат був далекий від ідеалу.

Комерційний зріст доповненої реальності є вражаючим. Їй, на відміну від віртуальної реальності, необов'язково спиратися на спеціалізоване залізо і громіздкі пристрої. Технологія прекрасно працює на смартфоні.

Доповнена реальність вже змінює наше сьогодення: віртуальні маски, полювання за покемонами по містах, діти, які стріляли один в одного не з деревинок, а через екран телефону. Зараз це вже реальність.

Наступний крок – масовий вихід доповненої реальності із зони розваг і соцмереж в сектор інформаційної підтримки. Автовиробники (поки лише Хендай, БМВ і Ауді, але список зростає) починають випускати додатки-доповнення до призначених для користувача інструкціям, що допомагають власникам наочно вивчити свій автомобіль. Все більше виробників техніки починають випускати додатки для ремонтних майстерень, які допомагають майстрам орієнтуватися у внутрішньому устрої складних приладів.

Побудова додатку на автоматичному обчисленні за допомогою векторних рівнянь, допоможе швидше і точніше проводити вимірювання за допомогою телефону. Ця програма корисна всім: кожна фізична компанія займається вимірюванням об'єктів, кожна людина стикається з необхідністю поміряти щось в особистих цілях. Так що цінністю цього додатка є не тільки сегмент великих фірм, а й простих користувачів.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

Людству довелося ближче познайомитися з технологією доповненої реальності (англ. Augmented reality, AR) в 2016 році, коли ігровий ринок мобільних додатків захопив Pokémon Go від компанії Niantic. З того моменту виник стереотип, що доповнена реальність – це цікава, але обмежена нова технологія, яка практично не виходить за межі мобільних ігор. Однак статистика 2021 року показує, що AR має свій вплив майже у всіх сучасних областях, від ігор до маркетингу, від навчання та моделювання до космічної інженерії.

Для початку, слід розібратися з термінологією доповненої реальності. Доповнена реальність (AR) – це поліпшена версія реального фізичного світу, яка досягається за рахунок використання цифрових візуальних елементів, звуку або інших сенсорних стимулів, що доставляються за допомогою технологій. Це швидко зростаюча тенденція серед компаній, що займаються мобільними обчисленнями і, зокрема, бізнес-додатками.

На тлі зростання збору і аналізу даних однією з основних цілей доповненої реальності є виявлення конкретних особливостей фізичного світу, поліпшення розуміння цих функцій і отримання розумних і доступних відомостей, які можна застосувати надалі до реальних фізичних додатків.

У міру того, як все більше галузей починають займатися AR і дізнаються, наскільки вона може бути корисною для їх реклами, людських ресурсів, досліджень і розробок, очікується, що в найближчому майбутньому практично кожний мобільний додаток буде використовувати цю технологію.

Адаптація до швидко мінливих трендам ринку вже відчутна і зараз; наприклад, Google Translate використовують AR для перекладу мов на льоту, просто навівши камеру смартфона на слово, сторінку книги або будь-яке інше зображення, що містить текст іноземною мовою. Quiver, що ще недавно займався

створенням книжок-розмальовок, виходить на новий рівень, перетворюючи кольорові зображення в тривимірних персонажів за допомогою AR. Доповнену реальність також можна знайти в сканерах QR-коду, зоряних картах, додатках для обміну миттєвими повідомленнями, онлайн-каталогах і т.п.

Однак, перш ніж розглядати майбутнє доповненої реальності, слід заглянути в її минуле. Технологія доповненої реальності була винайдена в 1968 році, коли Айвен Сазерленд, вчений в сфері програмної розробки і професор в Гарварді, створив перший в світі шолом з відображенням на дисплеї неіснуючих елементів, і назвав його «Дамоклів меч». Але термін «доповненої реальності» не використовувався до 1990 року, поки дослідник Том Кодел не почав його використовувати в своїх звітах при співпраці з корпорацією Boeing.

За 53 роки вивчення та поліпшення цієї технології, AR зазнав величезну кількість змін. До 1992 року люди не знали, як використовувати доповнену реальність без величезної машини віртуальної реальності (англ. Virtual reality, VR), спеціального шолома і рукавичок з датчиками. А зараз все, що потрібно — це наявність смартфона.

Але 53 роки – це досить довгий період часу. Повільний розвиток доповненої реальності має таке ж пояснення, чому і зараз AR не може поширюватися на ринку так швидко, як це пророкують. А саме:

а) обмежені можливості обладнання:

Мобільна доповнена реальність – це накладення якісних візуальних даних на навколишнє середовище, зняту на камеру користувача. Камери смартфонів мають різні можливості. Більшість з них можуть обробляти 2D-зображення, але не мають достатньо програмної та технічної підтримки для створення 3D-реальності. І незважаючи на те, що всі світові компанії випускають сучасну техніку з останніми існуючими компонентами, лише 56% власників смартфонів можуть використовувати AR.

Інша проблема пов'язана з точністю датчиків GPS. Потрібна висока точність даних, тому додатки можуть зіткнутися з проблемами коректного відображення інформації.

б) обмеження на взаємодії для багатьох користувачів:

Оскільки індустрія розробки AR все ще активно розвивається, часто виникають проблеми в експлуатації.

Додатки AR в даний час розробляються для використання на смартфонах, планшетах або інших портативних гаджетах. В результаті люди, які завантажують і використовують такі додатки, можуть отримати вигоду тільки з функціоналу для одного користувача.

У міру того, як AR стає доступнішою технологією, потреба доступу для багатьох користувачів, уніфікованого і оптимізованому інтерфейсу буде рости. Розробники повинні будуть знайти способи, що дозволяють забезпечити багатокористувацький інтерфейс, який був би досить оптимізований для роботи, незалежно від кількості учасників і навантаження.

Як і у випадку з обладнанням, компанії працюють над цією проблемою і вже досягли певного прогресу. Невидимий Поїзд (англ. The Invisible Train) — це гра з доповненою реальністю, розроблена командою з Грацького університету імені Карла Франценса. Його продуктивність оптимальна для 4 гравців.

в) проблеми з програмним забезпеченням:

Проблеми з технічним обладнанням – це лише одна сторона медалі. Розробники також стикаються з проблемами, пов'язаними з програмним забезпеченням.

Багато рішень, створені для розробників, з'явилися зовсім недавно. Два приклади – ARKit від Apple і ARCore від Google. І хоча на сьогоднішній день вони здаються досить перспективними і здатними спростити розробку додатків, їх потенціал до сих пір не повністю розкритий.

В основному перед розробниками стоїть завдання адаптувати нові комплекти для недавно випущеним пристроям. Багато хто з сучасних пристроїв оснащені вже власними технологіями. В результаті цього розробникам доводиться створювати додатки AR або для однієї платформи, або освоювати різні технології, кожна зі своїми особливостями.

Поточна інфраструктура додатків, на яку покладаються розробники, не відповідає вимогам по декільком іншим причинам. Наприклад, не існує способу інтеграції соціальних мереж з браузерами і додатками AR. Такі проблеми – нормальне явище для технологій, які ще перебувають на етапі початку розвитку. І хоча в найближчі роки очікується, що ситуація значно покращиться, розробникам важко підходити до розробки через проблеми, пов'язаних з програмним забезпеченням.

г) юридичні питання:

Юридична сторона, пов'язана з доповненою реальністю, вже давно є предметом обговорення, і очікується, що проблема стане ще більш складною в міру зростання масштабів впровадження технології.

Останнім часом особливо гостро постають питання конфіденційності та безпеки, особливо в зв'язку з появою нових правил стосовно цифрової інформації. Юристи обговорюють теми авторських прав і права власності. Наприклад, кому належить право на інформацію, яка була накладена на реальне зображення в додатку AR? В даний час не існує норм і стандартів, що застосовуються до розробки додатків AR і віртуальної реальності (VR). Власники доповнень можуть вступити в конфлікт з розробниками доповненої реальності, бажаючи володіти інформацією, яку програма надає користувачеві.

Крім того, існує ряд додатків, в якому користувачеві дають можливість створити власну реальність. Кому в такому випадку буде належати ідея? Чи буде воно власністю користувача або компанії, що створила цю програму? І хоча в даний час деякі з цих побоювань можуть здатися надуманими, вони можуть привести до тривалих і складних судових баталій в майбутньому.

У міру того, як AR стає все більш і більш поширеною, уряд і місцева влада будуть змушені приділяти їй особливу увагу. Смерть і збитки, завдані Pokémon Go, – лише один із прикладів того, чому в цій галузі необхідні вказівки і обмеження. Був навіть створений Pokémon Go Death Tracker з метою надання інформації про людей, які померли під час взаємодії з додатком доповненої реальності.

Зрештою, ці питання знайдуть своє рішення. Однак при цьому як і розробники, так і користувачі можуть втратити частину своїх прав.

Необхідність дотримуватися контролюючим нормам може стримувати зростання і розвиток цієї галузі.

Крім основних загальних проблем, пов'язаних з доповненою реальністю, слід зупинитися також на основних методах розробки (ARKit і ARCore), які в свою чергу також мають ряд проблем. Основними з цих проблем вважаються:

а) визначення кута камери:

Останнім часом в інтернеті дуже багато додатків, які дозволяють накласти неіснуючий віртуальний об'єкт поверх реального. Як приклад, маски доповненої реальності від Instagram – кожен користувач легко може створити будь-який об'єкт, який буде відображатися за допомогою камери. Це звучить досить легко, але і на сьогоднішній день з цією темою є деякі труднощі.

Додаток визначає площину, в якій повинен з'явиться віртуальний об'єкт, і правильно розміщує його в цій площині з урахуванням кута, під яким на нього дивиться користувач. І тут відразу виникає перша проблема: камера не надає інформацію про своє місцезнаходження, тобто програма не розуміє, з якого боку користувач дивиться на віртуальний об'єкт, на скільки градусів повернута камера і в якій частині об'єкт повинен бути показаний в даний момент.

Є кілька алгоритмів, які вирішують цю проблему. Наприклад, можна використовувати додатковий реальний об'єкт, відомий додатку (найчастіше використовують аркуш паперу А4 або монету). За допомогою розпізнавання цього додаткового об'єкта, виставляються точки на площині, за допомогою чого віртуальний об'єкт провертається в залежності від положення камери.

б) розпізнавання фізичних об'єктів:

Інша проблема – розпізнавання фізичного об'єкта. Припустимо, користувач хоче відсканувати об'єкт, наприклад, крісло. Після завершення сканування додаток визначає, що це крісло, а також визначає місце розташування користувача щодо об'єкта. А потім додаток може розмістити на ньому віртуальний об'єкт.

На цьому етапі виникає проблема визначення і сканування. У світі багато крісел і не може бути вибраний одне стандартне крісло, яке підходило б усім. Тобто, не може бути вибраний один певний шаблон для об'єкта. Як тільки можливо ідентифікувати речі, зовнішній вигляд яких сильно відрізняється один від одного?

В цьому випадку слід використовувати сучасні алгоритми штучного інтелекту.

в) визначення меж площині:

Камера смартфона показує двомірне зображення, і алгоритми можуть розпізнавати тільки деякі ключові точки на цьому зображенні. Однак визначення меж об'єктів і розташування фізичних і віртуальних об'єктів відносно один одного – надзвичайно складне завдання.

Наприклад, якщо помістити віртуальний об'єкт за фізичним об'єктом, його частина не повинна відображатися. Він повинен бути захований за фізичним об'єктом, а не створений поверх нього. Це називається оклюзією, коли частина фізичного об'єкта перекриває віртуальний об'єкт.

Різні компанії намагаються знайти рішення цієї проблеми, але на сьогоднішній день розробники не знайшли підходящу технологію.

1.2. Призначення розробки та галузь застосування

Розробка програмного забезпечення під керуванням ОС Android для виміру фізичніх розмірів об’єктів за допомогою камери мобільного телефону.

Для вимірювання фізичних об'єктів за допомогою камери найчастіше використовується спосіб математичного аналізу оточення: розробник задає початкові константи будь-якого предмета, який повинен фігурувати під час вимірювань. На основі цього камера захоплює двомірне зображення, порівнює вже знайомий предмет з необхідним об'єктом і надає результат. Проблема такого підходу полягає в неточності отриманих даних, а також необхідності надалі користувачеві завжди мати під рукою заданий початковий предмет.

Цей спосіб можна поліпшити за допомогою фізичного обчислення відстані камери до об'єкта за допомогою фреймворка OpenCV. Тоді користувачу необхідно задати параметри свого зросту, щоб визначити з якою приблизною висоти проводитися зйомка. Після чого необхідно зробити фотографії об'єкта з трьох ракурсів: з правої верхньої точки, лівої верхньої точки і з центральною. Незважаючи на те, що подібний метод є більш точним, він передбачає велику витрату сил для проведення необхідного обчислення. Крім того, мінусом двох цих способів є неможливість проводити вимірювання великих об'єктів, який може не поміщатися в межі камери.

Доповнена реальність – хоч і нова і досить перспективна технологія в сфері мобільної розробки, має нерозкритий потенціал. Використовуючи доповнену реальність для вимірювання фізичних об'єктів, можна не хвилюватися щодо меж камери, не ставити початкові розміри і не вказувати додаткові дані. В даному випадку, все, що знадобитися – це наявність телефону з Android-платформою.

1.3. Підстава для розробки

Відповідно до освітньої програми, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу.

Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується з наказом ректора.

Таким чином підставами для розробки (виконанням кваліфікаційної роботи) є:

– освітня програма спеціальності 121 «Інженерія програмного забезпечення»;

– навчальний план та графік навчального процесу;

– наказ ректора Національного технічного університету «Дніпровська політехніка» № 317-с від 07.06.2021 р;

– завдання на кваліфікаційну роботу на тему «Розробка програмного забезпечення під керуванням ОС Android для виміру фізичніх розмірів об’єктів за допомогою камери мобільного телефону».

1.4. Постановка завдання

Завданням даної роботи є написання мобільного додатку для автоматичного і спрощеного вимірювання фізичних реальних об'єктів за допомогою технології доповненої реальності. Основними характеристиками розробки повинні бути:

– можливість користувачеві самостійно робити вимірювання об'єкта (за допомогою проведення віртуальної рулетки);

– збереження та видалення отриманих даних;

– встановлення 3D моделей простих меблів та їх регулювання.

Поставлена задача може бути досягнута при виконанні наступних вимог:

– вивчення предметної області завдання;

– проведення порівняльної характеристики можливостей аналогічних програм;

– вибір платформи розробки;

– написання програмного коду;

– розробка довідника для пояснення користування додатком.

Кінцевим результатом має бути мобільний додаток, що розпізнає фізичні об'єкти різних розмірів, які можна виміряти за допомогою проведення віртуальної рулетки користувачем.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Кінцевий продукт повинен дотримуватися наступних функціональних вимог:

– інтуїтивно зрозумілий інтерфейс користувача;

– дані повинні вводитися користувачем на екрані;

– дані можуть зберігатися і видалятися користувачем.

1.5.2. Вимоги до інформаційної безпеки

Додаток запаковано в один файл, крім цього у користувача немає необхідності реєструватися і створювати аккаунт всередині програми. З цієї причини ніяких вимог до інформаційної безпеки немає.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для підтримки доповненої реальності на платформі Android, слід дотримуватися таким технічним вимогам:

– операційна система Android 8.1 (API 27) або вище;

– доступно не менше 1 ГБ вільного місця;

– зовнішня камера підтримує технологію віртуальної сцени.

Крім того, не всі марки телефонів мають внутрішню підтримку ARCore. Список всіх підтримуваних телефонів можна знайти на офіційному сайті Google. У листі часто зустрічаються такі бренди як Samsung, Motorola, Huawei, ASUS, Xiaomi, Google, Acer, HMD Global, LG, OnePlus, Oppo, Realme, Sony, Vivo та інші.

1.5.4. Вимоги до інформаційної та програмної сумісності

Технологія доповненої реальності (AR) дозволяє використовувати мобільні пристрої, щоб вибудовувати цифровий контент в реальний світ. Ця технологія, на відміну від віртуальної реальності (VR), не вимагає підключення гарнітур, окулярів і іншого додаткового устаткування. Все, що вам знадобитися, – це камера пристрою і додаток з функціями AR.

Для того, щоб технологія доповненої реальності коректно працювала і відображала весь функціонал додатка на пристрої Android, необхідно встановити на свій смартфон сервіси Google Play для AR, при цьому повинні бути дотримані наступні вимоги:

– виконано вхід в обліковий запис Google;

– є сертифікація ARCore.

Як основна мова програмування використовувалася Java.

Додаток було написано в Android Studio (4.1).

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Результатом даної кваліфікаційної роботи має бути мобільний додаток на Android з підтримкою технології доповненої реальності, для автоматичного обчислення розмірів фізичних об'єктів за допомогою камери смартфона.

Основний функціонал додатка полягає в умінні розпізнавати обраний користувачем об'єкт, який буде вимірюватися за допомогою проведення віртуальної лінії.

Крім цього, для кращої і якісної роботи програми, повинен бути присутнім наступний додатковий функціонал:

– можливість встановлення збережених 3D моделей (стіл, стілець, ліжко) за допомогою мобільної камери телефону;

– знімок екрану та зберігання зображення в галерею;

– збереження розрахунків в історію користувача.

2.2. Опис застосованих математичних методів

Мобільний додаток має будуватися на розрахунку дистанції між двома сферами, які визначаються користувачем. Головне завдання на етапі створення сфер, було розрахувати мінімальний радіус, відображений на рис.2.1., для комфортного використання програми, проте достатній, щоб можна було визначити початок і кінець бажаного відрізка.

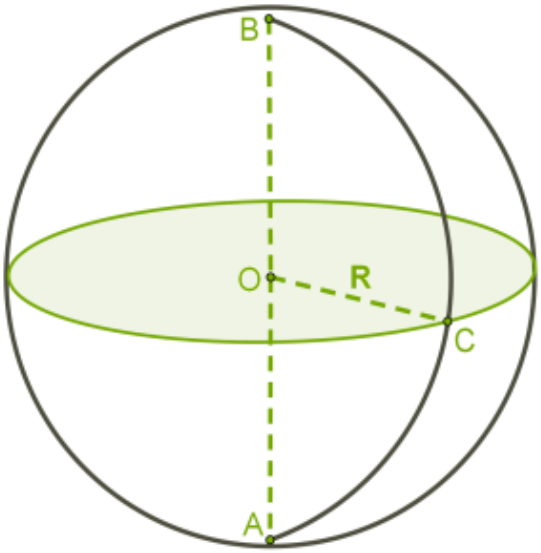

Рис.2.1. Зображення сфери та її радіусу

Для визначення бажаного радіуса був використаний практичний метод з використанням параметричного рівняння:

$$\begin{cases} x = x_0 + R \cdot \sin\theta \cdot \cos\varphi, \\ y = y_0 + R \cdot \sin\theta \cdot \sin\theta, \\ z = z_0 + R \cdot \cos\theta, \end{cases}$$

де $\theta \in [0, \pi], \varphi \in [0,2\pi]$

Для обчислення дистанції між сферами використовуються формули векторів, а саме:

– координати вектора: $\vec{\alpha}(x_a;\ y_a;\ z_a) \leftrightarrow \vec{\alpha} = x_a\vec{i} + y_a\vec{j} + z_a\vec{k}$;

– довжина вектора: $|\vec{\alpha}| = \sqrt{x_a^2 + y_a^2 + z_a^2}$;

2.3. Опис використаної архітектури та шаблонів проектування

Патерн проектування – це рішення певної проблеми, що часто зустрічається при проектуванні архітектури програм.

На відміну від готових функцій або бібліотек, патерн можна просто взяти і скопіювати в програму. Патерн є не якийсь конкретний код, а загальна концепція

вирішення тієї чи іншої проблеми, яку потрібно буде ще підлаштувати під потреби програми.

Самі низькорівневі і прості патерни – ідіоми. Вони не універсальні, оскільки застосовні тільки в рамках однієї мови програмування.

Самі універсальні – архітектурні патерни, які можна реалізувати практично на будь-якій мові. Вони потрібні для проектування всієї програми, а не окремих її елементів.

Крім того, патерни відрізняються і призначенням, наприклад:

а) породжувані патерни турбуються про гнучке створення об'єктів без внесення в програму зайвих залежностей;

б) структурні патерни показують різні способи побудови зв'язків між об'єктами;

в) поведінкові патерни піклуються про ефективну комунікацію між об'єктами.

Вивчаючи різні системи доповненої реальності, можна виявити, що більшість систем будуються на одній базовій архітектурної структурі, і відрізняються тільки в незначних деталях.

Крім того, однакові базові системи і підсистеми можуть бути знайдені в різних, абсолютно відмінних додатках. Це не дивно, адже доповнена реальність складається з інтерактивних з користувачем систем, і основний функціонал AR такий же, як і у всіх додатках з доповненою реальністю: відстеження положення користувача, змішування реальних і віртуальних об'єктів, обробка та відображення змін в залежності від дій користувача.

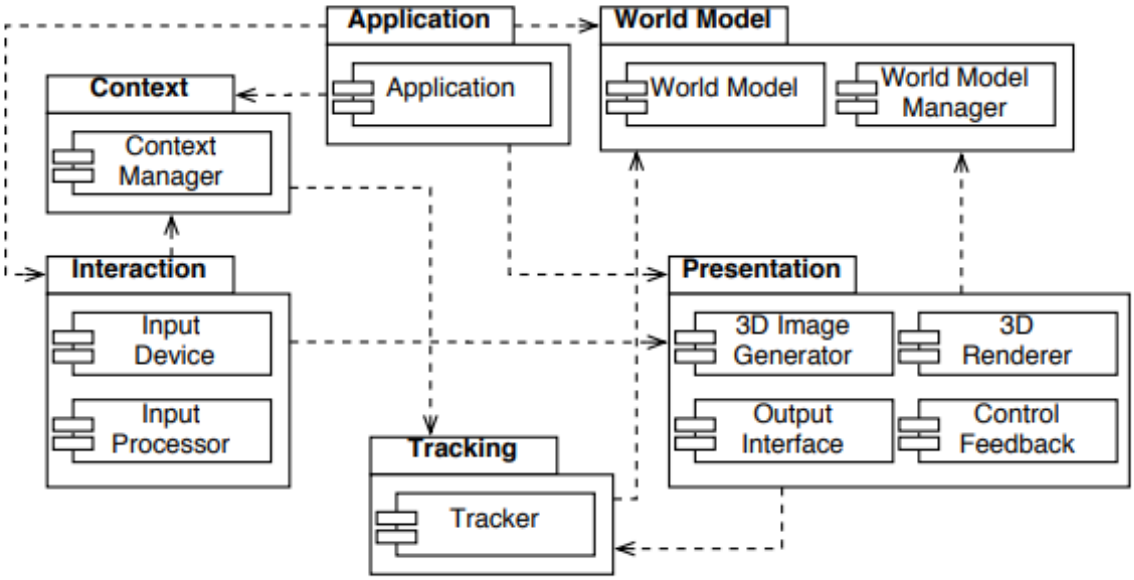

Рис.2.2. Типова модель архітектури для додатків доповненої реальності

Ця схожість основного функціоналу серед різних систем AR дозволяє розподілити основну систему на шість різних підсистем. Кожна підсистема передбачає частковий функціонал всієї системи.

Така абстрактна структура, яку використовують для додатків доповненої реальності, дуже схожа з Model-View-Controller (MVC, «Модель-Представлення-Контролер»). MVC шаблон розділяє інтерактивні системи на «Модель», «Представлення» і «Контролер» для контролю інформацією. «Модель» зберігає дані додатку, дає доступ до методів, обробляє введення інформації з контролера; «Представлення» відображає призначену для користувача інформацію, оновлення та зміни в моделі, створює «Контролер»; «Контролер» пов'язаний з «Представленням», дає доступ до методів введення користувача, пересилає події в «Модель», і ініціалізує зміни в «Представленні». В області досліджень human-computer interaction (HCI, «людино-комп’ютерна взаємодія) були запропоновані альтернативні терміни для даного шаблону: так, замість «Представлення» розробники доповненої реальності називають це «Презентацією» (Presentation), а замість «Контролера» – Інтеракція (Interaction). Схожість цих структур не є дивною, адже системи доповненої реальності за своїм визначенням інтерактивні.

2.4. Опис використаних технологій та мов програмування

Для відображення віртуальних об'єктів повинна створюватися віртуальна площина з опорними точками, на якій можна розмістити необхідні об'єкти. Існує кілька основних напрямків AR, які визначають динаміку розвитку і подальшої розробки програми:

а) «маркерна» технологія;

б) «безмаркерна» технологія;

в) комбінована («просторова») технологія.

AR технологія на базі спеціальних маркерів, або міток, зручна тим, що вони простіше розпізнаються камерою і дають їй більше жорстку прив'язку до місця для віртуальної моделі. Така технологія набагато надійніше «безмаркерної» і працює практично без збоїв. Під маркером розуміється об'єкт, розташований в навколишньому просторі, який перебуває і аналізується спеціальним програмним забезпеченням для подальшого відтворення віртуальних об'єктів. Найчастіше в ролі маркера виступає аркуш паперу з деяким спеціальним зображенням. Тип малюнка може варіюватися досить сильно і залежить від алгоритмів розпізнавання зображень.

Безмаркерні технології часто застосовуються в мобільних додатках, і будуються за допомогою спеціальних датчиків: акселерометр, гіроскоп, магнетометр, GPS-приймач. Це новий, але дуже динамічно прогресуючий напрямок, в основі якої лежать особливі алгоритми розпізнавання, за допомогою яких на навколишній ландшафт, знятий камерою, накладається віртуальна «сітка». На цій сітці програмні алгоритми знаходять опорні точки, за якими визначають точне місце розташування віртуального об’єкту.

Перевага такої технології полягає в тому, що об'єкти реального світу, тобто будь-які зображення служать маркерами самі по собі і для них не потрібно створювати спеціальних візуальних ідентифікаторів, як це робиться у випадку з маркерною технологією.

Крім маркерної і безмаркерної, існує технологія доповненої реальності, заснована на просторовому розташуванні об'єкта. У ній використовуються дані GPS, гіроскопа і компаса, вбудованого в мобільний телефон. Місце віртуального об'єкта визначається координатами в просторі. Активація програми доповненої реальності відбувається при збігу координати, закладеної в програмі, з координатами користувача.

В даному проекті використовується комбінована технологія, так як на основі побудованої «сітки», користувач може визначити в просторі дві необхідні точки.

Архітектурний шаблон програмного забезпечення описує конкретну проблему проекту, яка виникає при створенні і надалі експлуатації додатки, а також надає загальну схему вирішення цієї проблеми. Схема рішення визначає задіяні компоненти проекту і їх взаємини між собою. На початковому етапі для визначення логіки програми були використані технології «низькорівневого графічного примітивного шаблону» і «слідкуючого сервера».

Низькорівневий графічний примітивний шаблон (Low-level Graphics Primitives pattern) – це нова методика використання 3D-бібліотек для побудови примітивних тривимірних об'єктів. У разі даного проекту, за допомогою цього методу були побудовані сфери, що відображають початок і кінець потрібної користувачеві дистанції. Крім цього, низькорівневий графічний примітивний шаблон дає можливість відстежувати стан і дистанцію побудованих об'єктів, зв'язуючись з відслідковувати сервером.

Слідкуючий сервер (Tracking Server) дозволяє зчитувати інформацію, введену з боку користувача і повернути результат після проведених над даними операціями. Оптичне відстеження є основою доповненої реальності.

Хоча в ніші AR використовується цілий ряд мов програмування, серед яких C#, C/C ++, JavaScript, Swift та навіть Python, Java є одним з найбільш затребуваних. Причини, власне, все ті ж, що і у випадку з іншими технологіями: Java-код дозволяє програмам працювати швидко, об'єктно-орієнтованість полегшує створення окремих компонентів додатків, а платформонезалежність робить AR продукти на Java більш універсальними.

У випадку з доповненою реальністю Java є одною з основних мов програмування, тому що абсолютна більшість AR-додатків створюються для мобільних пристроїв. Відповідно, Java є основним для AR-додатків, що створюються для пристроїв на базі мобільної операційної системи Android. Тому у випадку з даним проектом, додаток повністю написано на Java.

Крім того, для програми були використані набір засобів для розробки програмного забезпечення (SDK), випущені від Google – ARCore. Основу роботи інструментарію становлять камери і зовнішні датчики, оскільки доповнена реальність базується на горизонтальних поверхнях і інших оцифрованих об'єктах реального світу. ARCore перш за все має такі можливості:

– поділяти світ на умовні суті, знаходити об'єкти, визначати їх розміри та інші параметри;

– відслідковувати рух, в тому числі розрізняти цілеспрямоване переміщення пристрою і випадкову тряску;

– сприймати освітленість, знаходити джерела світла, розрізняти тіні;

– визначати джерела звуку, розпізнавати голоси, особи, жести і так далі.

Концептуально, в основі ARCore лежить 3 основних положення – motion tracking («відстеження руху»), environment understanding («зчитування навколишнього середовища») і light estimation («оцінка світла»).

Коли мобільний телефон переміщається по всьому світу, ARCore об'єднує візуальні дані з камери пристрою, щоб оцінити становище і орієнтацію об'єктива щодо часу і простору. Це і називається motion tracking («відстеження руху»).

У цій категорії слід виділити різні калібрування. Оптичне калібрування, pinholde model («модель камери з точковим отвором») – математична модель, що описує відношення між координатами точки в тривимірному просторі з її проекцією на полотно, а також Field of View (FoV, «поле зору») – модель описує спотворення перспективи зображення; фотометричне калібрування – карта інтенсивності кольорів і моделювання на основі інерції тощо.

Втім, існує температурна проблема при заводському калібруванні IMU («інерційний вимірювальний пристрій»). Різні виробники виготовляють її при різній температурі, тому дані на різних девайсах можуть відрізнятися.

Що стостується environment understanding («зчитування навколишнього середовища»), то в цьому випадку ARCore шукає кластери характерних точок, які, як здається, лежать на загальних горизонтальних поверхнях і роблять їх доступними для застосування в вигляді площин. Приклад побудови можна побачити на рис.2.3.

В основі розуміння оточення лежить технологія SLAM – simultaneous localization and mapping («одночасна локалізація та картографування»). SLAM карта є графіком 3D точок, які представляють собою розріджену хмару, де кожна позначка відповідає координатам оптичного об'єкта (наприклад, кут таблиці).

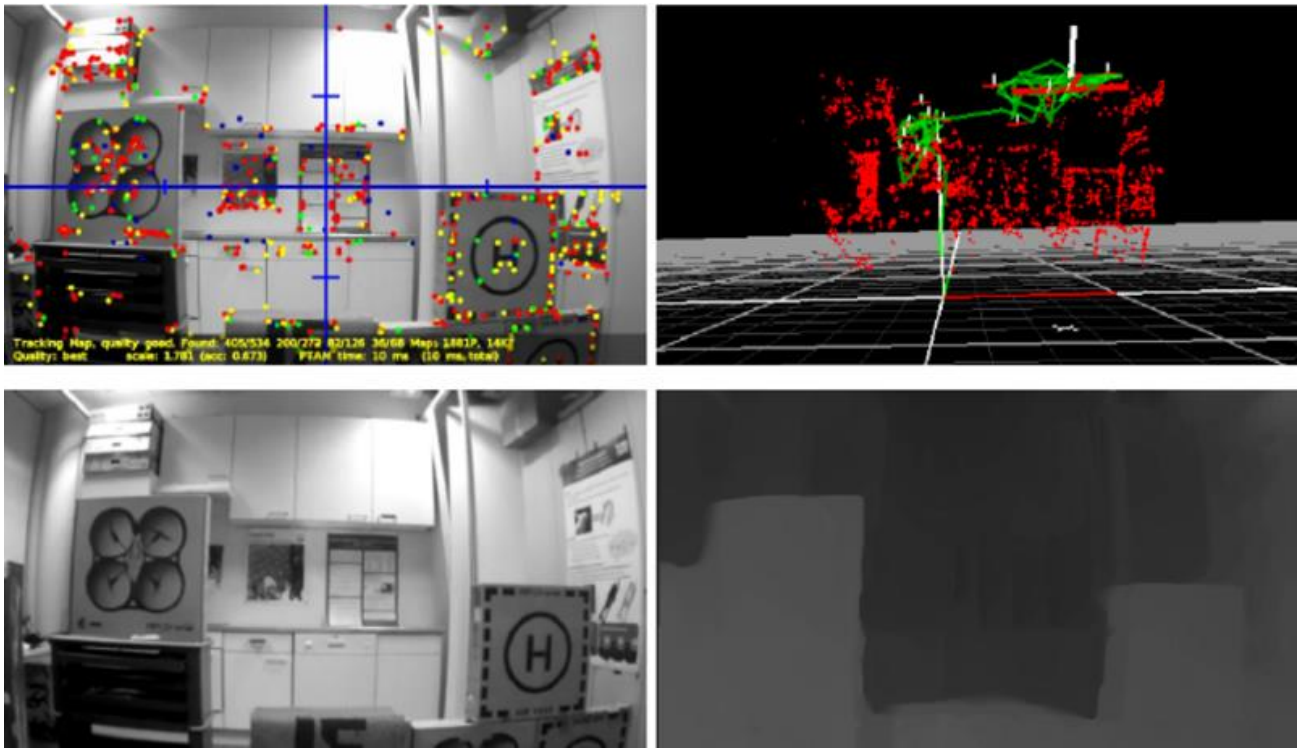

Рис.2.3. Побудування 3D точок

Також як і з вимірами на основі прискорення, SLAM спирається на карту точок, які можуть бути більш-менш надійними.

Основне завдання SLAM – побудова і оновлення карти невідомого середовища при одночасному відстеження місця розташування користувача всередині неї. Також як і Tango, має проблеми з дзеркальними поверхнями.

І останнім положенням є light estimation («оцінка світла»).

ARCore може виявити середню інтенсивність зображення камери, так що користувач отримує можливість висвітлити віртуальні об'єкти так само, як освітлені об'єкти навколишнього середовища.

2.5. Опис структури програми та алгоритми її функціонування

Система доповненої реальності без точкової реєстрації зображення не буде підтримуватися в багатьох додатках. Тому одна з найважливіших проблем в доповненої реальності, яку потрібно доопрацювати, є розширення зіставлення уявлень, тобто зв'язування різних сцен координат і координат об'єктів із загальними координатами. Більш того, щоб досягти максимально ефективний результат доповненої реальності слід оцінити камеру і її можливість розпізнавати положення об'єкта. Дані для аналізу зазвичай беруться із закладених текстурних даних або збираються з датчиків.

Один з найбільш часто використовуваних підходів до аналізу зображення при отриманні необхідний даних є маркери, що забезпечують орієнтири і відмінні елементи відповідно до яких об'єкти можуть бути ідентифіковані. Крім того, маркери є важливим інструментом через їх високий ступінь точності захоплення. Система також повинна вміти обчислювати положення камери за допомогою виявленого маркера. Буде досить розпізнати чотири точки, щоб дізнатися, як розташовується камера. Крім того, маркери дозволяють точно визначати координати навколишнього середовища і накладати поверх локальні додаткові об'єкти.

З цієї причини першорядна умова, при якому система доповненої реальності буде працювати правильно – це ефективність алгоритму розпізнавання маркерів. При правильному ідентифіковані маркерів виходить дізнатися відстань від

одного маркера до іншого, кути повороту телефону і рівень освітлення навколишнього середовища.

Для опису алгоритму розпізнавання маркерів, в першу чергу, слід розглянути типи використовуваних маркерів і їх ідентифікаційні дані.

Для доповненої реальності можна використовувати різні типи маркерів. Найбільш часто використовуваними є: стовпці (наприклад, QR, DataMatrix, PDF-417), кола (наприклад, CircleInner, CircleSplit, CircleOuter, SpotCode), квадрати (ідентифікатори маркерів, виконавчі маркери) і зображення (наприклад, StudierStube).

Найчастіше використовується квадратний маркер з товстої рамкою, так як він є найбільш точним у визначенні. Це відбувається через:

– форми маркера – легко виявити, навіть використовуючи інші методи або алгоритми;

– кольоровий контраст – навіть при різному сприйнятті кольору камерами, маркер буде виявлятися;

– тип кордону – найбільш надійний для виявлення.

Алгоритм розпізнавання маркера оснований на мові програмування Java, але підтримується і на інших мовах, таких як C#, і на різних операційних системах (наприклад, Android), що підтримують віртуальні машини.

Основною перевагою використання саме цього алгоритму є вирішення проблеми відстеження точки з боку користувача. Алгоритм дозволяє відстежувати кут огляду без втрати віртуального об'єкта. Це забезпечує чітке зображення без втрати якості.

Алгоритм починається з введення базового зображення. Після цього, під час завантаження списку компонентів, запускається визначення меж зображення. При завершенні цього етапу, додаток самостійно шукає заготовлені шаблони і проводить аналіз при зіставленні знайденого зображення з шаблоном.

Все це робиться тільки за даними кутів. Після застосування шаблону, камера налаштовується за отриманими вказівками для фінального кроку. У разі успішного розпізнавання маркер відображає введене зображення.

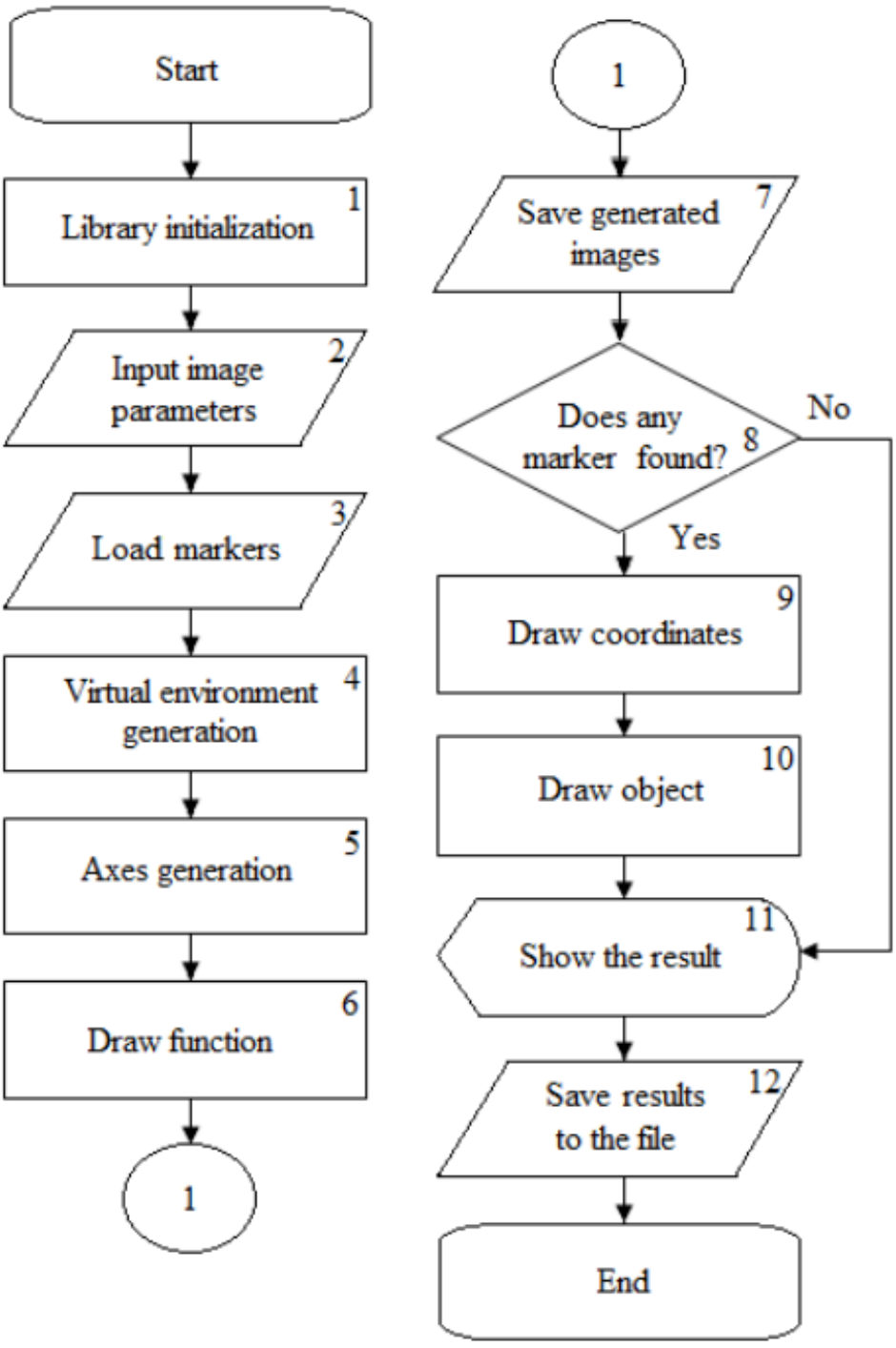

Рис.2.4. Блок-схема імплементації системи доповненої реальності

Розглядаючи схему використання та імплементації доповненої реальності, можна звернути увагу що на третьому етапі завантажується близько ста маркерів. Така кількість є мінімальним для чіткого побудови сітки і розпізнавання об'єкта.

На наступних кроках створюється віртуальне середовище, куди поміщаються створені маркери. Після чого йде розпізнавання цих маркерів.

Слід розглянути структуру мобільного додатка.

Додаток запускається з MainActivity, основними завданнями якого є:

– надання меню користувача з чотирма кнопками («Виміряти», «Встановити», «Галерея», «Вихід»);

«Виміряти» – перехід до класу MeasureActivity.

«Встановити» – перехід до класу LocateActivity.

«Галерея» – в разі існуючих збережених зображень у ході програми перехід до класу GalleryActivity, в іншому випадку відображення тексту «Збережених зображень не існує».

«Вихід» означає завершення роботи програми і вихід користувача на головну сторінку телефону.

– відображення імені розробника і версії програми.

Результат роботи меню MainActivity представлений нижче.

На рис.2.5. відображена робота верхньої навігаційної панелі. За допомогою кнопки «Developer» (з анг. «Розробник») на екрані з’являється ім’я розробника мобільного додатку (рис.2.6.), а кнопки «Version» (з англ. «Версія») – остання версія додатку відповідно (рис.2.7).

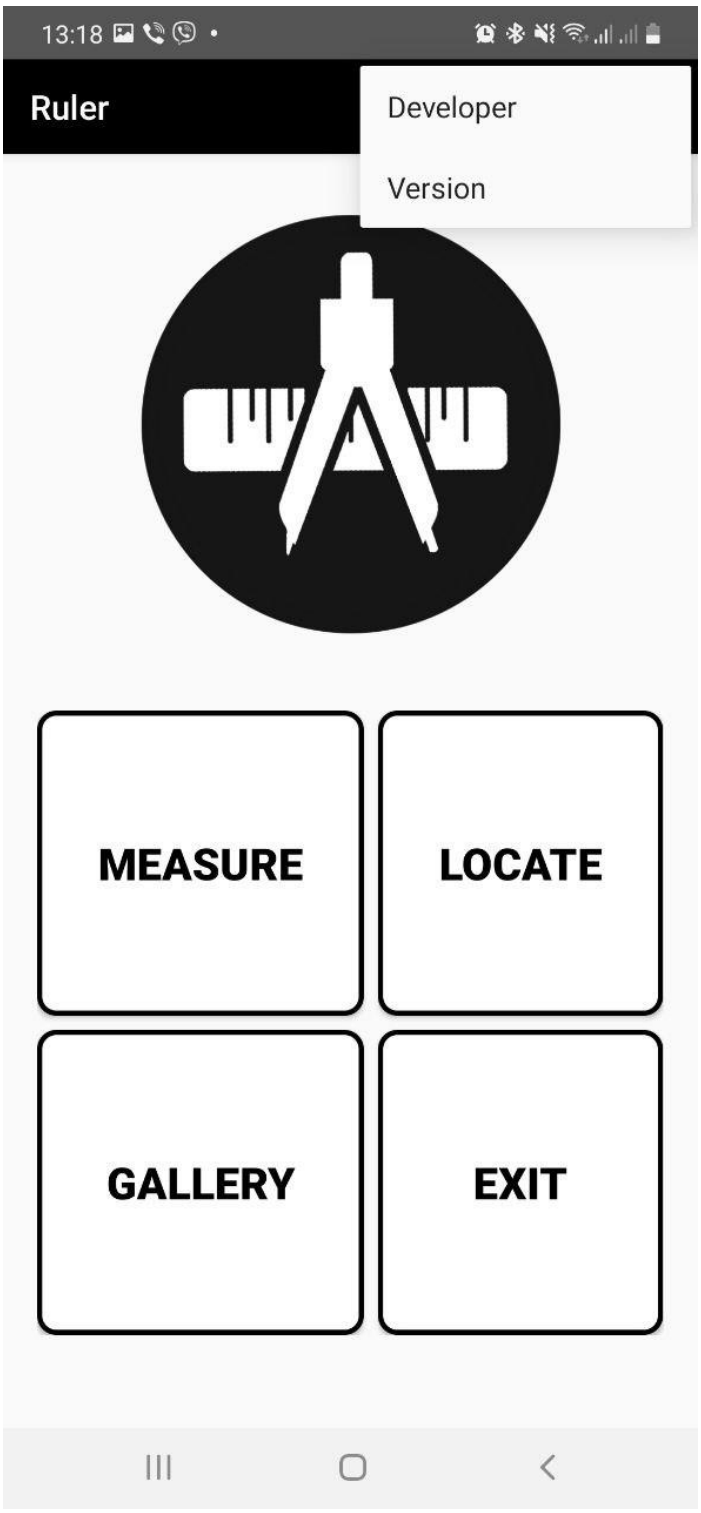

Рис.2.5. Головне меню з натисканням на додаткову вкладку

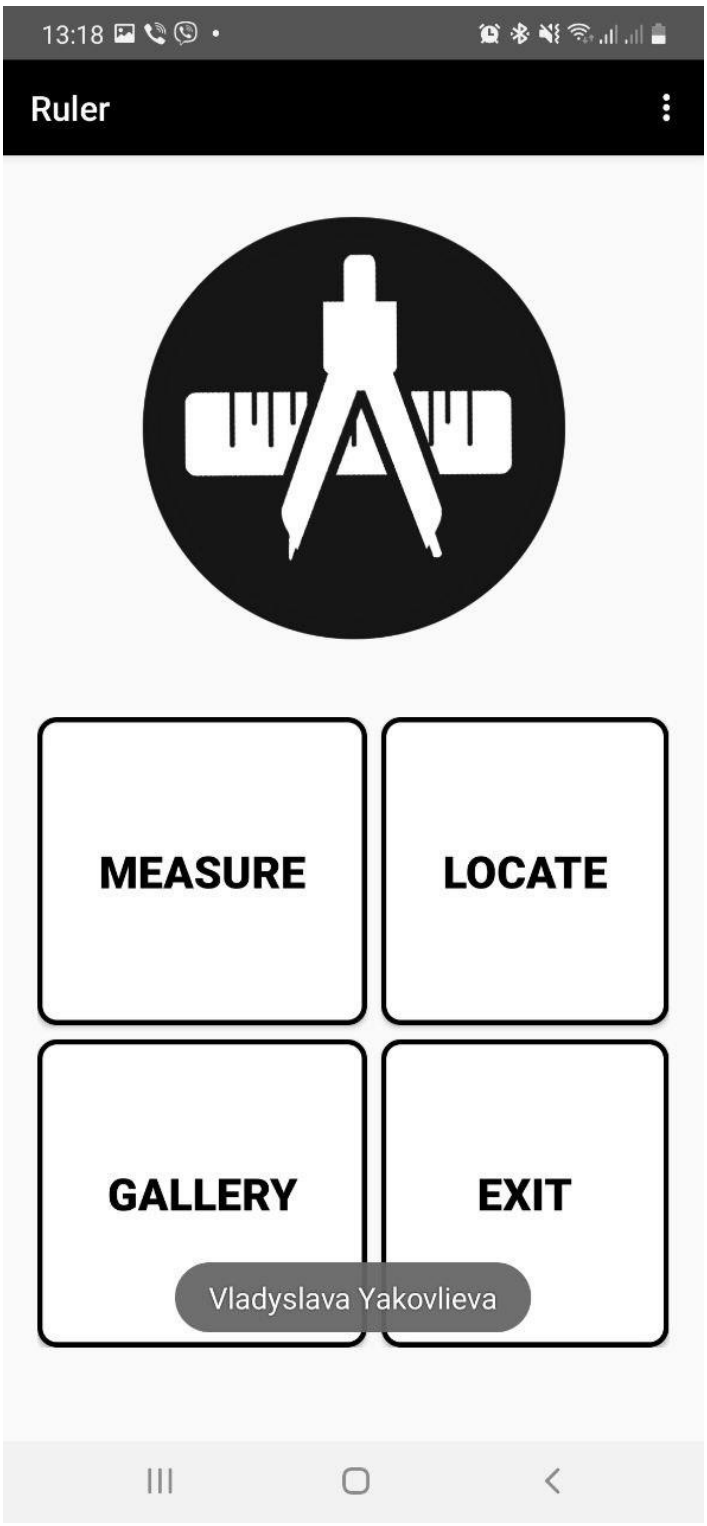

Рис.2.6. Результат нажаття кнопки «Розробник»

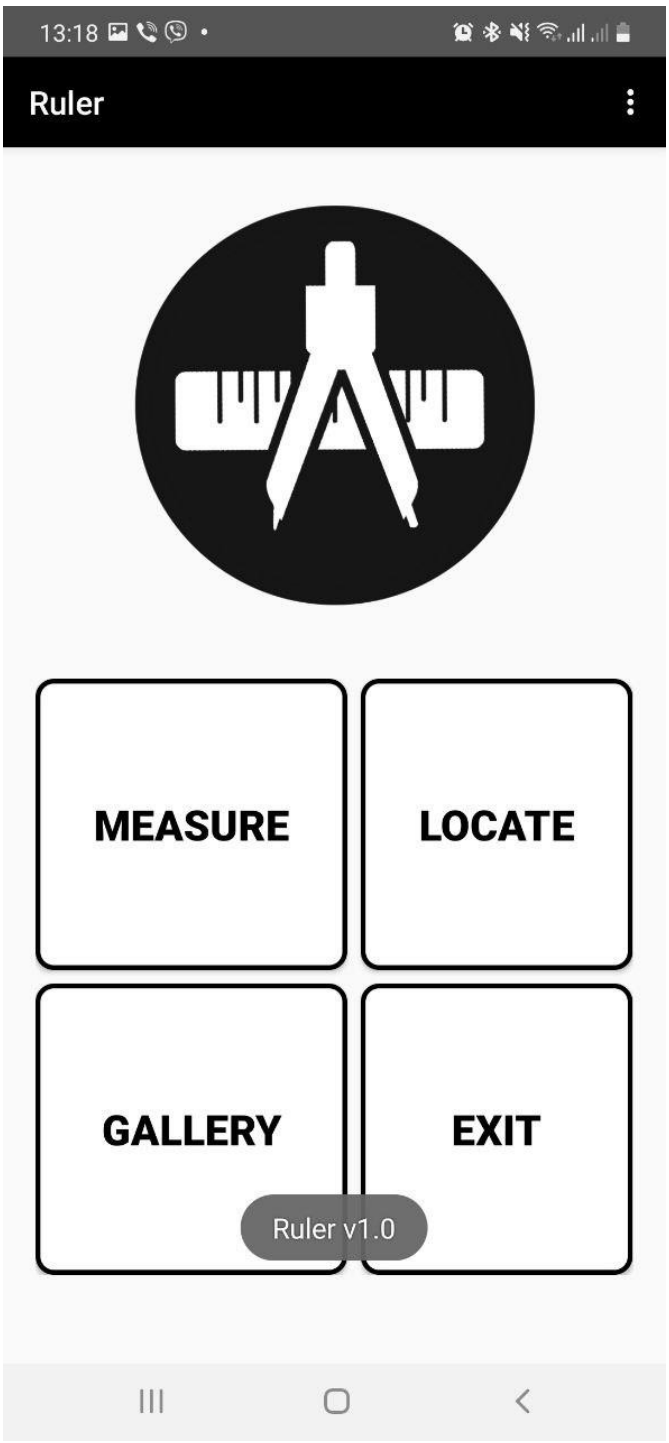

Рис.2.7. Результат нажаття кнопки «Версія»

MeasureActivity створює простір для подальшої побудови користувальницької лінії. Основними завданнями цього класу є:

– перевірка на сумісність телефону з ARCore, а також на можливі помилки, пов'язані з неправильно встановленими додатковими сервісами:

```
if (!mSession.isSupported(mConfig)) {
    Log.d(TAG, "This device is not support ARCore.");
```

– зчитування початкової і кінцевої точки користувальницької лінії:

```
mLastX = event.getX();
mLastY = event.getY();
mPointAdded = true;
```

– видалення встановленних точок:

```
if (!mPoints.isEmpty()) {
   mPoints.remove(mPoints.size() - 1);
   mRenderer.removePoint();
   updateDistance();
```

– розрахунок довжини побудованої лінії:

```
double distance = Math.sqrt(
      (start[0] - end[0]) * (start[0] - end[0])
            + (start[1] - end[1]) * (start[1] - end[1])
            + (start[2] - end[2]) * (start[2] - end[2]));
totalDistance += distance;
```

LocateMeasure дозволяє користувачеві встановить вже створені шаблонні 3D-об'єкти. Головними завданнями даного класу є:

– вибір та встановленния (рис.2.8.) одного з трьох об'єктів (стіл, стілець, ліжко);

– можливість зміни розмірів і положення об'єктів в просторі;

– функція збереження зображення в «Галерею».

Рис.2.8. Результат встановлення об’єкту «Стілець»

GalleryActivity дає можливість користувачеві переглянути збережені в процесі користування додатком зображення. Крім того, існує можливість видалення цих зображень.

2.6. Обґрунтування та організація вхідних та вихідних даних програми

Механізм введення представляє собою матеріальний пристрій для прямої взаємодії користувача з доповнюючими віртуальними об'єктами. Основна проблема механізмів введення полягає в семантичному розриві між реальними і віртуальними об'єктами, що негативно впливає на простоту сприйняття інтерфейсу користувачем.

Тривимірний віртуальний об'єкт має ступені свободи, отже, те ж число ступенів свободи повинен мати керуючий вхідний сигнал для здійснення повного контролю над цим об'єктом. Послати такий сигнал здатний як специфічний пристрій введення, розроблений для конкретної розв'язуваної задачі, так і стандартний периферійний пристрій на зразок клавіатури. Проте, спосіб введення ближчий до виробленої маніпуляції з віртуальним об'єктом, наприклад, жестовий, створює менше когнітивне навантаження на користувача, ніж введення за допомогою буквено-цифрової клавіатури. Це означає, що для роботи з більш розвиненим механізмом введення потрібна менша кваліфікація користувача. У той же час розробка і реалізація більш досконалих різновидів механізмів введення вимагає значно більших витрат, а самі механізми пов'язані з нестандартними пристроями введення.

За цією логікою простими і, отже, найменш інтуїтивно зрозумілими механізмами введення є двомірне введення за допомогою стандартних периферійних пристроїв введення: клавіатура, миша і імітація контролера за допомогою персональних ЕОМ. На рівень вище розташовуються фізичні просторові контролери, наприклад, джойстики, робота з якими більш наочно відображає маніпуляцію віртуальними об'єктами. Найкращий за дією над віртуальним об’єктом – жестовий механізм, мається на увазі звернення з візуалізованою інформацією, як з матеріальної сутністю, а також матеріальне введення використовує спрощену фізичну модель віртуального об'єкта для управління ним. Саме такий варіант був задіяний в мобільній додатку, що розробляється в рамках даної кваліфікаційної роботи.

Механізми виведення в системах доповненої реальності поділяються на візуальні, акустичні, тактильні тощо. Головна функція механізму виведення – відображення користувальницьких сценаріїв і завдань на призначені для цього реальних об'єктів. Так як функціонал проекту не передбачає більш, ніж відображення інформації на дисплеї, то і вивід використовувався візуальний.

2.7. Опис розробленого програмного продукту

Кожна програма на Android складається з безлічі різних файлів, як в даному випадку, дванадцять функціональних класів, один файл з певними константами і шаблони завантажених зображень для дизайну програми. Щоб додаток міг коректно встановлюватися і працювати, необхідно правильно упакувати всі складові частини в один великий архівний файл, який називається APK (Android Package Kit). Для установки файлу APK не потрібно нічого, крім самого смартфона або планшета на Android.

У ході тестування мобільного додатку було виявлено, що сервіси доповненої реальності від Google можуть працювати некоректно в залежності від версії операційної системи та потужності внутрішніх компонентів. Так, з двох реальних мобільних смартфонів, Samsung Galaxy A50, куплений у 2020 році, працював досить швидко, без помилок, відображаючи близькі до реальних фізичних значень за допомогою задовільної якості камери. Ulefone S7 був куплений у 2017 році. Сервіси доповненої реальності працювали некоректно, камера відображала значення з неприпустимою похибкою, швидкість обробки запросів смартфона була дуже низька.

2.7.1. Використані технічні засоби

При розробці програми була використана персональна ЕОМ з наступними характеристиками:

– Процесор Intel Core i7-8750H;

– Відеопроцесор Nvidia GeForce GTX 1060 (6 ГБ GDDR5);

– Оперативна пам'ять 16 ГБ DDR4-2666.

Тестування проводилося на таких смартфонах:

– Samsung Galaxy A50;

– Nexus 6;

– Ulefone S7.

2.7.2. Використані програмні засоби

Додаток було написано в Android Studio (4.1) – в середовищі розробки на основі IntelliJ IDEA, що надає інтегровані інструменти для розробки та налагодження додатків для платформі Android. У процесі тестування був також використаний внутрішній емулятор з віртуальним смартфоном Nexus 6.

2.7.3. Виклик та завантаження програми

Після загрузки файлу APK, необхідно пройти усі етапи встановлення додатку на смартфон за рекомендаціями від Android. Наступним етапом буде запуск додатку на смартфоні чи планшеті.

2.7.4. Опис інтерфейсу користувача

Інтерфейс додатку є стандартним та зручним для розуміння користувачу будь-якого рівня. Після запуску програми, користувач має змогу бачити головне меню додатку (рис.2.9). Увесь функціонал додатку є інтуїтивно зрозумілим і не передбачає спеціалізованих знань у даній сфері. Окрім того, для кращого розуміння роботи додатку, користувач може прочитати коротку інструкцію з користування.

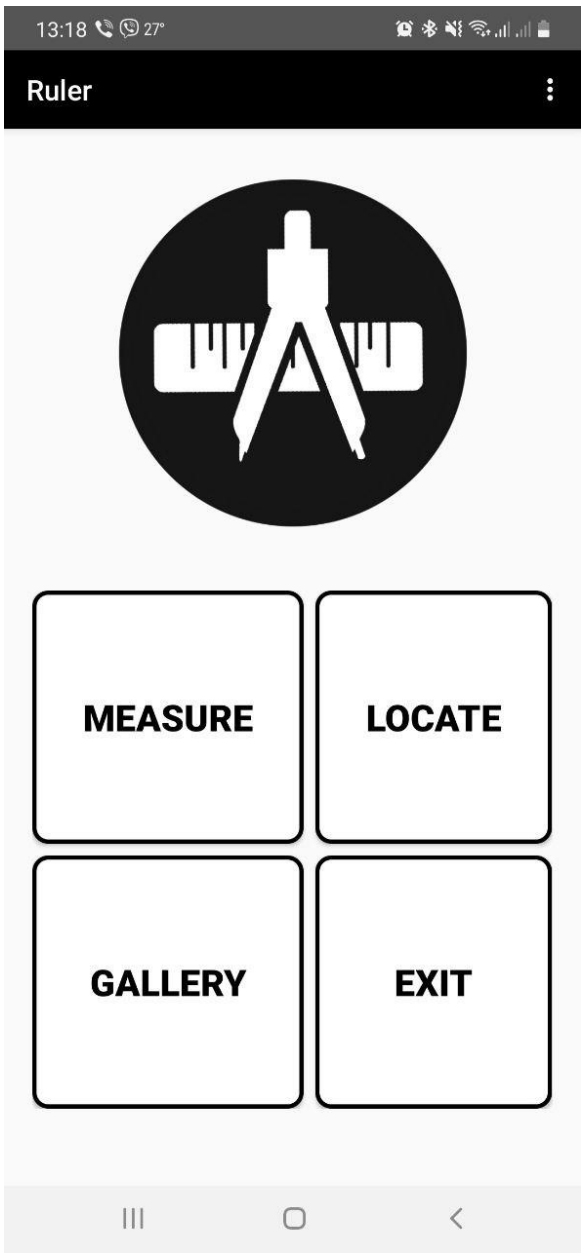

Рис.2.9. Головне меню мобільного додатку

Рис.2.10. Меню вимірювань об'єкта

При практичному вимірі об'єкта (стільця), довжина становила 44,5 сантиметра (рис.2.10). Похибка становить 0,5 сантиметра.

Для простішого розуміння інтерфейсу додатку було встановлене лише зберігання та відображення фотографій, які були зроблені у самому додатку. Мобільний додаток немає дозволу на перегляд інших зображень, що робить

користування додатком не тільки більш зручним, але й безпечним через заборону додатку редагувати внутрішні файли. Внутрішня галерея зображена на рис.2.11.

Рис.2.11. Галерея та фото з встановленим 3D об’єктом

РОЗДІЛ 3
ЕКОНОМІЧНІЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів програми – 1256;
2. коефіцієнт складності програми – 1,3;
3. коефіцієнт корекції програми в ході її розробки – 0,07;
4. годинна заробітна плата програміста – 112 грн/год;
5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,1;
7. вартість машино-години ЕОМ – 13 грн/год.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д \text{, людино-годин,} \qquad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_a – витрати праці на розробку блок-схеми алгоритму;

t_n – витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ;

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p), \qquad (3.2)$$

де q – передбачуване число операторів (1256);

C – коефіцієнт складності програми (1,3);

p – коефіцієнт кореляції програми в ході її розробки (0,07).

$$Q = 1256 \cdot 1{,}3 \cdot (1 + 0{,}07) = 1747;$$

Витрати праці на вивчення опису задачі ti визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75 \ldots 85) \cdot K}, \text{ людино-годин} \qquad (3.3)$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі (1,2);

K – коефіцієнт кваліфікації програміста, обумовлений стажем роботи з даної спеціальності (1,1);

$$t_u = \frac{1747 \cdot 1{,}2}{85 \cdot 1{,}1} = 22{,}4, \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20 \ldots 25) \cdot K}; \qquad (3.4)$$

$$t_a = \frac{1747}{20 \cdot 1{,}1} = 79{,}4, \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)\cdot K}; \quad (3.5)$$

$$t_n = \frac{1747}{25 \cdot 1{,}1} = 63{,}5, \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{\text{отл}} = \frac{Q}{(4...5) \cdot K}; \quad (3.6)$$

$$t_{\text{отл}} = \frac{1747}{5 \cdot 1{,}1} = 317{,}63, \text{ людино-годин,}$$

– за умови комплексного налагодження завдання:

$$t^{\text{к}}_{\text{отл}} = 1{,}2 \cdot t_{\text{отл}}; \quad (3.7)$$

$$t^{\text{к}}_{\text{отл}} = 1{,}2 \cdot 317{,}63 = 381{,}16, \text{ людино-годин,}$$

Витрати праці на підготовку документації:

$$t_{\partial} = t_{\partial p} + t_{\partial o}; \quad (3.8)$$

де $t_{\partial p}$ – трудомісткість підготовки матеріалів і рукопису;

$$t_{\partial} = \frac{Q}{(15...20)\cdot K}; \quad (3.9)$$

$$t_{\partial} = \frac{1747}{20 \cdot 1{,}1} = 79{,}4, \text{ людино-годин,}$$

де $t_{до}$ – трудомісткість редагування, печатки й оформлення документації;

$$t_{до} = 0{,}75 \cdot t_{др}; \qquad (3.10)$$

$$t_{до} = 0{,}75 \cdot 79{,}4 = 59{,}55, \text{ людино-годин.}$$

$$t_{д} = 79{,}4 + 59{,}55 = 138{,}95, \text{ людино-годин.}$$

Отримаємо трудомісткість розробки програмного забезпечення:

t = 50 + 22,4+ 79,4 + 63,5 + 317,63 + 138,95 = 671,88, людино-годин.

У результаті ми розрахували, що в загальній складності необхідно 671,88 людино-годин для розробки даного програмного забезпечення.

3.2. Рахунок витрат на створення програми

Витрати на створення ПЗ *Кпо* включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн,} \qquad (3.11)$$

$З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн,} \qquad (3.12)$$

де t – загальна трудомісткість, людино-годин;

$С_{ПР}$ – середня годинна заробітна плата програміста, грн/година.

З урахуванням того, що середня годинна зарплата програміста становить 112 грн/год, то отримаємо:

$$З_{ЗП} = 671{,}88 \cdot 112 = 75250{,}56, \text{ грн.}$$

Вартість машинного часу $З_{МВ,}$ необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{МВ} = t_{отл} \cdot C_{М}, \text{ грн,} \qquad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год;

$C_{МЧ}$ – вартість машино-години ЕОМ, грн/год.

$$З_{МВ} = 317{,}63 \cdot 13 = 4129{,}19 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$К_{ПО} = 75250{,}56 + 4129{,}19 = 79379{,}75 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ мес.} \qquad (3.14)$$

де B_k – число виконавців;

$F_р$ – місячний фонд робочого часу (при 40 годинному робочому тижні Fр=176 годин).

Витрати на створення программного продукту:

$$T = \frac{671{,}88}{1 \cdot 176} = 3{,}8 \text{ міс.}$$

Висновки. Мобільний додаток має вартість 79379,75 грн. Ймовірний очікуваний час розробки – 3,8 місяці при стандартному 40-годинному робочому тижні і 178-годинному робочому місяці. Цей термін пов’язаний з кількістю

операторів і включає в себе час для дослідження та розробку алгоритму розв’язання задачі, розробку дизайну і створення документації. На розробку мобільного додатку буде витрачено 671,88 людино-годин.

ВИСНОВКИ

Метою кваліфікаційної роботи бакалавра є розробка мобільного додатку на Android, що вимірює фізичні розміри об’єктів за допомогою камери телефону. В процесі написання кваліфікаційної роботи було встановлено, що мобільний додаток буде корисним як і для бізнесу будь-якого напрямку, так і для індивідуальних користувачів для особистих цілей.

Додаток має додатковий функціонал для спрощення користуванням, а саме:

– збереження та видалення отриманих даних;

– встановлення 3D моделей простих меблів та їх регулювання.

Актуальність поставленох задачі обумовлюється широким попитом на такі програмні продукти, що використовують технологію доповненої реальності. На сьогоднішній день мобільна розробка Є однією з найбільш популярних напрямків в інформаційних технологіях. КРМО того, доповнена реальність - тренд останніх років, який тільки підвищує інтерес до даного проекту.

Програма працює під керуванням Android, яка широко використовується цільовою аудиторією продукту. Додаток реалізований на мові програмування Java з використанням набору засобів розробки програмного забезпечення (SDK) – ARCore. Мобільний додаток заснований на комбінованій технології (маркерної та безмаркерної) з застосуванням алгоритму розпінавання маркеру.

В «Економічному розділі» визначено трудомісткість розробки програмного забезпечення (671,88 чол-год), підраховані витрати на створення програмного забезпечення (79379,75 грн.) і гаданий період розробки (3,8 міс.).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. R. T. Azuma. A Survey of Augmented Reality, Presence. – №6. –1997. – 385 p.
2. F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stall. Pattern-Oriented Software Architecture / F. Buschmann // A System of Patterns, John-Wiley & Sons. –1996. –130 p.
3. A. Butz, T. Hollerer, S. Feiner, B. Macintyre, C. Beshers. Enveloping Users and Computers in a Collaborative 3D Augmented Reality. – 2016. – 44 p.
4. S. Feiner, B. Macintyre, T. Holler. A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment. – 2015. – 50 p.
5. S. Feiner, B. Macintyre, D. Seligmann. Knowledge-based augmented reality, Communications of the ACM. – Berlin: 2015. – 62 p.
6. W. Freidrich, W. Wolhgemuth. Augmented Reality for Development, Production and Service, in The International Workshop on Potential Industrial Applications of Mixed and Augmented Reality. –2019. –P. 430-432.
7. B. Macintyre, S. Feiner. Language-level support for exploratory programming of distributed virtual environments. –2017. –105 p.
8. C. Owen, A. Tang, F. Xiao. A Blended Script and Compiled Code Development Systems for Augmented Reality. –2018. –244 p.
9. W. Piekarski, B. Thomas. An Object Oriented Software Architecture for 3D Mixed Reality Applications. –2016. –45 p.
10. T. Reicher, A. Macwilliams.Study on Software Architectures for Augmented Reality Systems. –2015. –210 p.
11. X. Zhang, S. Fronz and N. Navab, Visual Marker Detection and Decoding in AR Systems. –2018. – P. 97–106.
12. M. Mahvash, L. Besharati Tabrizi, A novel augmented reality system of image projection for image-guided neurosurgery. –2019. –943 p.

13. AR Measure turns your phone into a virtual measuring tape. URL: https://www.curbed.com/2017/6/29/15894556/armeasure-app-augmented-reality-ruler-measuring-tape-ios. Last Accessed: 04.05.2021.

14. What is Augmented Reality? URL: https://www.fi.edu/what-is-augmented-reality. Last Accessed: 22.02.2021.

15. C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. –IEEE Transactions on Robotics. – 2016. –8 p.

16. Что такое ARCore? Всё, что вам нужно знать. URL: https://devcolibri.com/what-is-arcore/. Дата звернення: 18.02.2021.

17. Google ARCore: Androids's augmented reality tech explained. URL: https://www.pocket-lint.com/ar-vr/news/google/142054-google-arcore-android-s-equivalent-to-apple-arkit-explained. Last Accessed: 03.04.2021.

18. Android Manifest: Specifying Android App and SDK Versions. URL: https://howtodoinjava.com/android/android-manifest-specifying-android-app-and-sdk-versions/. Last Accessed: 14.03.2021.

19. Android Augmented Reality – Android ARCore Example. URL: https://www.journaldev.com/21479/android-augmented-reality-arcore-example. Last Accessed: 20.03.2021.

20. How to build an Augmented Reality Android App with ARCore and Android Studio. URL: https://www.freecodecamp.org/news/how-to-build-an-augmented-reality-android-app-with-arcore-and-android-studio-43e4676cb36f/. Last Accessed: 25.02.2021.

21. Augmented Reality in Mobile Apps Technology. URL: http://zesium.com/powerful-augmented-reality-in-mobile/. Last Accessed: 01.06.2021.

22. Android: Augmented reality (AR) app that overlays 3D object in the scene. URL: http://www.anandmuralidhar.com/blog/android/simple-ar/. Last Accessed: 03.06.2021.

23. Как готовить AR на андроиде. URL: https://habr.com/ru/post/347140/. Дата звернення: 19.02.2021.

24. How to Use React Native & OpenCV for Image Processing. URL: https://brainhub.eu/library/opencv-react-native-image-processing/. Last Access: 15.05.2021.

25. Working with the OpenCV Camera for Android: Rotating, Orienting, and Scaling. URL: https://heartbeat.fritz.ai/working-with-the-opencv-camera-for-android-rotating-orienting-and-scaling-c7006c3e1916. Last Access: 18.04.2021.

ДОДАТОК А

КОД ПРОГРАМИ

AndroidManifest.xml //файл описує важливу інформацію про інструменти збірки

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.arcore.ruler">

    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-feature android:name="android.hardware.camera.ar" android:required="true"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>

    <application
        android:allowBackup="true"
        android:icon="@drawable/ruler_logo"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"
        tools:ignore="GoogleAppIndexingWarning">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity android:name=".MeasureActivity"
            android:label="@string/activity_measure_label">
        </activity>

        <activity android:name=".LocateActivity"
            android:label="@string/activity_locate_label">
        </activity>

        <activity android:name=".GalleryActivity"
            android:label="@string/activity_gallery_label">
        </activity>
        <activity android:name=".ManualActivity"
            android:label="Manual">
        </activity>
        <meta-data android:name="com.google.ar.core" android:value="required"/>
        <provider
            android:name="android.support.v4.content.FileProvider"
            android:authorities="com.ruler.fileprovider"
            android:exported="false"
            android:grantUriPermissions="true">

            <meta-data
                android:name="android.support.FILE_PROVIDER_PATHS"
                android:resource="@xml/filepaths" />
        </provider>

    </application>

</manifest>
```

Java

CameraRender //файл, в якому прописується методи вузіалізації камери

```
package com.arcore.ruler;

import android.opengl.GLES11Ext;
import android.opengl.GLES20;
import android.util.Log;

import com.google.ar.core.Frame;

import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.nio.FloatBuffer;

public class CameraRenderer {
   private static final String TAG = CameraRenderer.class.getSimpleName();

   private final String vertexShaderString =
         "attribute vec4 aPosition;\n" +
         "attribute vec2 aTexCoord;\n" +
         "varying vec2 vTexCoord;\n" +
         "void main() {\n" +
         "   vTexCoord = aTexCoord;\n" +
         "   gl_Position = aPosition;\n" +
         "}";

   private final String fragmentShaderString =
         "#extension GL_OES_EGL_image_external : require\n" +
         "precision mediump float;\n" +
         "uniform samplerExternalOES sTexture;\n" +
         "varying vec2 vTexCoord;\n" +
         "void main() {\n" +
         "    gl_FragColor = texture2D(sTexture, vTexCoord);\n" +
         "}";

   private static final float[] QUAD_COORDS = //створення точок координат
         new float[] {-1.0f, -1.0f, 0.0f, -1.0f, 1.0f, 0.0f, 1.0f, -1.0f, 0.0f, 1.0f, 1.0f, 0.0f};

   private static final float[] QUAD_TEXCOORDS =
         new float[] {0.0f, 1.0f, 0.0f, 0.0f, 1.0f, 1.0f, 1.0f, 0.0f};

   private static final int COORDS_PER_VERTEX = 3;
   private static final int TEXCOORDS_PER_VERTEX = 2;

   private int[] mTextures;
   private FloatBuffer mVertices;
   private FloatBuffer mTexCoords;
   private FloatBuffer mTexCoordsTransformed;
   private int mProgram;

   public CameraRenderer() {
      mVertices = ByteBuffer.allocateDirect(QUAD_COORDS.length * Float.SIZE /
8).order(ByteOrder.nativeOrder()).asFloatBuffer();
      mVertices.put(QUAD_COORDS);
      mVertices.position(0);

      mTexCoords = ByteBuffer.allocateDirect(QUAD_TEXCOORDS.length * Float.SIZE /
8).order(ByteOrder.nativeOrder()).asFloatBuffer();
      mTexCoords.put(QUAD_TEXCOORDS);
      mTexCoords.position(0);

      mTexCoordsTransformed = ByteBuffer.allocateDirect(QUAD_TEXCOORDS.length * Float.SIZE /
8).order(ByteOrder.nativeOrder()).asFloatBuffer();
   }
```

```
    public void init() {
        mTextures = new int[1];
        GLES20.glGenTextures(1, mTextures, 0);
        GLES20.glBindTexture(GLES11Ext.GL_TEXTURE_EXTERNAL_OES, mTextures[0]);
        GLES20.glTexParameteri(GLES11Ext.GL_TEXTURE_EXTERNAL_OES, GLES20.GL_TEXTURE_WRAP_S,
GLES20.GL_CLAMP_TO_EDGE);
        GLES20.glTexParameteri(GLES11Ext.GL_TEXTURE_EXTERNAL_OES, GLES20.GL_TEXTURE_WRAP_T,
GLES20.GL_CLAMP_TO_EDGE);
        GLES20.glTexParameteri(GLES11Ext.GL_TEXTURE_EXTERNAL_OES,
GLES20.GL_TEXTURE_MIN_FILTER, GLES20.GL_NEAREST);
        GLES20.glTexParameteri(GLES11Ext.GL_TEXTURE_EXTERNAL_OES,
GLES20.GL_TEXTURE_MAG_FILTER, GLES20.GL_NEAREST);

        Log.d(TAG, "[EDWARDS] texture id : " + mTextures[0]);

        int vShader = GLES20.glCreateShader(GLES20.GL_VERTEX_SHADER);
        GLES20.glShaderSource(vShader, vertexShaderString);
        GLES20.glCompileShader(vShader);
        int[] compiled = new int[1];
        GLES20.glGetShaderiv(vShader, GLES20.GL_COMPILE_STATUS, compiled, 0);
        if (compiled[0] == 0) {
            Log.e(TAG, "Could not compile vertex shader.");
            GLES20.glDeleteShader(vShader);
        }

        int fShader = GLES20.glCreateShader(GLES20.GL_FRAGMENT_SHADER);
        GLES20.glShaderSource(fShader, fragmentShaderString);
        GLES20.glCompileShader(fShader);
        GLES20.glGetShaderiv(fShader, GLES20.GL_COMPILE_STATUS, compiled, 0);
        if (compiled[0] == 0) {
            Log.e(TAG, "Could not compile fragment shader.");
            GLES20.glDeleteShader(fShader);
        }

        mProgram = GLES20.glCreateProgram();
        GLES20.glAttachShader(mProgram, vShader);
        GLES20.glAttachShader(mProgram, fShader);
        GLES20.glLinkProgram(mProgram);
        int[] linked = new int[1];
        GLES20.glGetProgramiv(mProgram, GLES20.GL_LINK_STATUS, linked, 0);
        if (linked[0] == 0) {
            Log.e(TAG, "Could not link program.");
        }
    }

    public void draw() { //представлення знайдених точок на площині
        GLES20.glBindTexture(GLES11Ext.GL_TEXTURE_EXTERNAL_OES, mTextures[0]);

        GLES20.glUseProgram(mProgram);

        int position = GLES20.glGetAttribLocation(mProgram, "aPosition");
        int texcoord = GLES20.glGetAttribLocation(mProgram, "aTexCoord");

        GLES20.glVertexAttribPointer(position, COORDS_PER_VERTEX, GLES20.GL_FLOAT, false, 0, mVertices);
        GLES20.glVertexAttribPointer(texcoord, TEXCOORDS_PER_VERTEX, GLES20.GL_FLOAT, false, 0,
mTexCoordsTransformed);

        GLES20.glEnableVertexAttribArray(position);
        GLES20.glEnableVertexAttribArray(texcoord);

        GLES20.glDrawArrays(GLES20.GL_TRIANGLE_STRIP, 0, 4);

        GLES20.glDisableVertexAttribArray(position);
```

```
        GLES20.glDisableVertexAttribArray(texcoord);
    }

    public int getTextureId() {
        return mTextures[0];
    }

    public void transformDisplayGeometry(Frame frame) {
        frame.transformDisplayUvCoords(mTexCoords, mTexCoordsTransformed);
    }
}
```

GalleryActivity

```
package com.arcore.ruler;

import android.annotation.SuppressLint;
import android.content.Context;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.os.Environment;
import android.support.v4.content.FileProvider;
import android.support.v7.app.AppCompatActivity;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.Gallery;
import android.widget.ImageView;
import android.widget.Toast;

import java.io.File;

public class GalleryActivity extends AppCompatActivity {

    File selectFile;
    int currentPos;
    boolean viewClickSwitch=true;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_gallery);
        setTitle("Gallery");

    }

    public class MyGalleryAdapter extends BaseAdapter {
        Context context;
        File[] imgFiles;

        public MyGalleryAdapter  (Context c, File[] Files) {
            context = c;
            imgFiles = Files;
        }
```

```
    @Override
    public int getCount() {
        return imgFiles.length;
    }

    @Override
    public Object getItem(int position) {
        return null;
    }

    @Override
    public long getItemId(int position) {
        return 0;
    }

    @SuppressLint("ClickableViewAccessibility")
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        ImageView imageView = new ImageView(context);
        imageView.setLayoutParams(new Gallery.LayoutParams(108, 204));
        imageView.setScaleType(ImageView.ScaleType.FIT_CENTER);
        imageView.setPadding(0,15,0,0);

        if(imgFiles[position].exists()) {
            Bitmap myBitmap = BitmapFactory.decodeFile(imgFiles[position].getAbsolutePath());
            imageView.setImageBitmap(myBitmap);
        }

        final int pos = position;
        imageView.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                ImageView picPreview = (ImageView) findViewById(R.id.PicPreview);
                picPreview.setScaleType(ImageView.ScaleType.FIT_CENTER);

                if(imgFiles[pos].exists()) {
                    Bitmap myBitmap = BitmapFactory.decodeFile(imgFiles[pos].getAbsolutePath());
                    picPreview.setImageBitmap(myBitmap);
                    viewClickSwitch=true;
                }

                selectFile = imgFiles[pos];
                currentPos = pos;

                picPreview.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        if(viewClickSwitch){
                            sendFile(imgFiles[pos]);
                        }
                    }
                });

                return false;
            }
        });
        return imageView;
    }
}

public boolean onCreateOptionsMenu(Menu menu){
    super.onCreateOptionsMenu(menu);
```

```
        MenuInflater galleryInflater = getMenuInflater();
        galleryInflater.inflate(R.menu.gallery_option, menu);
        return true;
    }

    public boolean onOptionsItemSelected(MenuItem item){
        switch (item.getItemId()){
            case R.id.delete:
                selectFile.delete();

                ImageView picPreview = (ImageView) findViewById(R.id.PicPreview);
                picPreview.setScaleType(ImageView.ScaleType.FIT_CENTER);
                picPreview.setImageBitmap(null);
                viewClickSwitch=false;
                Toast.makeText(getApplicationContext(), "Deleted.", Toast.LENGTH_SHORT).show();

                onResume();
                return true;
        }
        return false;
    }

    public void sendFile(File file){
        Uri uri = null;
        if (Build.VERSION.SDK_INT > Build.VERSION_CODES.M) {
            uri = FileProvider.getUriForFile(GalleryActivity.this, "com.ruler.fileprovider", file);
        }else{
            uri = Uri.fromFile(file);
        }
        Intent intent = new Intent(Intent.ACTION_SEND); //спосіб відправлення зображення
        intent.setType("image/*"); //визначення збереженого файлу як jpg
        intent.putExtra(Intent.EXTRA_STREAM, uri);
        startActivity(intent); //Виклик зображення
    }

    @Override
    protected void onResume() {
        super.onResume();

        File[] imageFiles;
        imageFiles = new File(Environment.getExternalStorageDirectory().getAbsolutePath()+"/Ruler").listFiles();

        Gallery gallery = (Gallery) findViewById(R.id.Gallery1);
        MyGalleryAdapter galAdpater = new MyGalleryAdapter(this, imageFiles);
        gallery.setAdapter(galAdpater);

    }
}
```

Line //клас побудови лінії

```
package com.arcore.ruler;

import android.graphics.Color;
import android.opengl.GLES20;
import android.opengl.Matrix;
import android.util.Log;

import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.nio.FloatBuffer;
import java.nio.ShortBuffer;
```

```
public class Line {

    private static final String TAG = Line.class.getSimpleName();

    public static final int RED = 0;
    public static final int GREEN = 1;
    public static final int BLUE = 2;
    public static final int ALPHA = 3;

    private final String vertexShaderString =
            "attribute vec3 aPosition;\n" +
            "attribute vec4 aColor;\n" +
            "uniform mat4 uMvpMatrix; \n" +
            "varying vec4 vColor;\n" +
            "void main() {\n" +
            "  vColor = aColor;\n" +
            "  gl_Position = uMvpMatrix * vec4(aPosition.x, aPosition.y, aPosition.z, 1.0);\n" +
            "}";

    private final String fragmentShaderString =
            "precision mediump float;\n" +
            "varying vec4 vColor;\n" +
            "void main() {\n" +
            "  gl_FragColor = vColor;\n" +
            "}";

    private boolean mIsInitialized = false;

    private int mProgram;

    private float[] mModelMatrix = new float[16];
    private float[] mViewMatrix = new float[16];
    private float[] mProjMatrix = new float[16];

    private FloatBuffer mVertices;
    private FloatBuffer mColors;
    private ShortBuffer mIndices;

    private float mLineWidth = 1.0f;

    public Line(float startX, float startY, float startZ, float endX, float endY, float endZ, int lineWidth, int color) {
        float[] vertices = new float[]{ startX, startY, startZ, endX, endY, endZ };

        float r  = Color.red(color) / 255.f;
        float g = Color.green(color) / 255.f;
        float b = Color.blue(color) / 255.f;
        float a = Color.alpha(color) / 255.f;

        float[] colors = new float[]{ r, g, b, a, r, g, b, a };

        short[] indices = new short[] { 0, 1 };

        mVertices = ByteBuffer.allocateDirect(vertices.length * Float.SIZE /
8).order(ByteOrder.nativeOrder()).asFloatBuffer();
        mVertices.put(vertices);
        mVertices.position(0);

        mColors = ByteBuffer.allocateDirect(colors.length * Float.SIZE /
8).order(ByteOrder.nativeOrder()).asFloatBuffer();
        mColors.put(colors);
        mColors.position(0);

        mIndices = ByteBuffer.allocateDirect(indices.length * Short.SIZE /
```

```
8).order(ByteOrder.nativeOrder()).asShortBuffer();
        mIndices.put(indices);
        mIndices.position(0);

        mLineWidth = (float) lineWidth;
    }

    public void init() {
        int vShader = GLES20.glCreateShader(GLES20.GL_VERTEX_SHADER);
        GLES20.glShaderSource(vShader, vertexShaderString);
        GLES20.glCompileShader(vShader);
        int[] compiled = new int[1];
        GLES20.glGetShaderiv(vShader, GLES20.GL_COMPILE_STATUS, compiled, 0);
        if (compiled[0] == 0) {
            Log.e(TAG, "Could not compile vertex shader.");
            GLES20.glDeleteShader(vShader);
        }

        int fShader = GLES20.glCreateShader(GLES20.GL_FRAGMENT_SHADER);
        GLES20.glShaderSource(fShader, fragmentShaderString);
        GLES20.glCompileShader(fShader);
        GLES20.glGetShaderiv(fShader, GLES20.GL_COMPILE_STATUS, compiled, 0);
        if (compiled[0] == 0) {
            Log.e(TAG, "Could not compile fragment shader.");
            GLES20.glDeleteShader(fShader);
        }

        mProgram = GLES20.glCreateProgram();
        GLES20.glAttachShader(mProgram, vShader);
        GLES20.glAttachShader(mProgram, fShader);
        GLES20.glLinkProgram(mProgram);
        int[] linked = new int[1];
        GLES20.glGetProgramiv(mProgram, GLES20.GL_LINK_STATUS, linked, 0);
        if (linked[0] == 0) {
            Log.e(TAG, "Could not link program.");
        }

        mIsInitialized = true;
    }

    public void draw() { //відображення лінії користувача
        GLES20.glUseProgram(mProgram);

        int position = GLES20.glGetAttribLocation(mProgram, "aPosition");
        int color = GLES20.glGetAttribLocation(mProgram, "aColor");
        int mvp = GLES20.glGetUniformLocation(mProgram, "uMvpMatrix");

        float[] mvMatrix = new float[16];
        float[] mvpMatrix = new float[16];
        Matrix.multiplyMM(mvMatrix, 0, mViewMatrix, 0, mModelMatrix, 0);
        Matrix.multiplyMM(mvpMatrix, 0, mProjMatrix, 0, mvMatrix, 0);

        GLES20.glUniformMatrix4fv(mvp, 1, false, mvpMatrix, 0);

        GLES20.glEnableVertexAttribArray(position);
        GLES20.glVertexAttribPointer(position, 3, GLES20.GL_FLOAT, false, 4 * 3, mVertices);

        GLES20.glEnableVertexAttribArray(color);
        GLES20.glVertexAttribPointer(color, 4, GLES20.GL_FLOAT, false, 4 * 4, mColors);

        GLES20.glLineWidth(mLineWidth);
        GLES20.glDrawElements(GLES20.GL_LINES, mIndices.capacity(), GLES20.GL_UNSIGNED_SHORT,
mIndices);
```

```
        GLES20.glLineWidth(1.0f);

        GLES20.glDisableVertexAttribArray(position);
    }

    public boolean isInitialized() {
        return mIsInitialized;
    }

    public void setModelMatrix(float[] modelMatrix) {
        System.arraycopy(modelMatrix, 0, mModelMatrix, 0, 16);
    }

    public void setProjectionMatrix(float[] projMatrix) {
        System.arraycopy(projMatrix, 0, mProjMatrix, 0, 16);
    }

    public void setViewMatrix(float[] viewMatrix) {
        System.arraycopy(viewMatrix, 0, mViewMatrix, 0, 16);
    }
}
```

LocateActivity

```
package com.arcore.ruler;

import android.app.Activity;
import android.hardware.display.DisplayManager;
import android.opengl.GLSurfaceView;
import android.opengl.Matrix;
import android.os.Bundle;
import android.util.Log;
import android.view.Display;
import android.view.GestureDetector;
import android.view.MotionEvent;
import android.view.ScaleGestureDetector;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import com.google.ar.core.ArCoreApk;
import com.google.ar.core.Camera;
import com.google.ar.core.Config;
import com.google.ar.core.Frame;
import com.google.ar.core.HitResult;
import com.google.ar.core.Plane;
import com.google.ar.core.PointCloud;
import com.google.ar.core.Pose;
import com.google.ar.core.Session;
import com.google.ar.core.Trackable;
import com.google.ar.core.TrackingState;
import com.google.ar.core.exceptions.CameraNotAvailableException;
import com.google.ar.core.exceptions.UnavailableApkTooOldException;
import com.google.ar.core.exceptions.UnavailableArcoreNotInstalledException;
import com.google.ar.core.exceptions.UnavailableDeviceNotCompatibleException;
import com.google.ar.core.exceptions.UnavailableSdkTooOldException;
import com.google.ar.core.exceptions.UnavailableUserDeclinedInstallationException;
```

```
import java.util.Collection;
import java.util.List;
import java.util.Locale;
import java.util.Timer;
import java.util.TimerTask;

public class LocateActivity extends Activity {
    private static final String TAG = LocateActivity.class.getSimpleName();
    private boolean mUserRequestedInstall = true;

    private TextView locate_rotate;
    private TextView locate_scale;


    private TextView mTextView;
    private GLSurfaceView mSurfaceView;

    private MainRenderer mRenderer;

    private Session mSession;
    private Config mConfig;

    //Змінні, що відображають обрані координати
    private float mCurrentX;
    private float mCurrentY;

    private float mScaleFactor = 0.02f;
    private float mRotateFactor = 0.0f;

    private final int TABLE = 0;
    private final int CHAIR = 1;
    private final int BED = 2;

    //Визначає положення об’єкту
    //Змінна, яка визначає який було обрано об’єкт
    private int mSelectedModel = -1;

    private float[] mModelMatrix = new float[16];

    private float[] mTableModelMatrix = new float[16];
    private float[] mChairModelMatrix = new float[16];
    private float[] mBedModelMatrix = new float[16];

    private boolean[] mModelInit = { false, false, false };
    private boolean[] mModelPut = { false, false, false }; //Перевірка на розміщення об’єкту


    //Змінна перемикача, яка активується при подвійному натисканні
    private boolean mIsPut = false;

    private GestureDetector mGestureDetector; //Змінна положення меблів
    private ScaleGestureDetector mScaleDetector; //Змінна величини меблів

        private Button btn_capture_locate;

        //save Check
        private Boolean isSaveClick = false;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            hideStatusBarAndTitleBar();
            setContentView(R.layout.activity_locate);
```

```
mTextView = (TextView) findViewById(R.id.txt_locate);
mSurfaceView = (GLSurfaceView) findViewById(R.id.gl_surface_view);

locate_rotate = (TextView) findViewById(R.id.locate_rotate);
locate_scale = (TextView) findViewById(R.id.locate_scale);


final DisplayManager displayManager = (DisplayManager) getSystemService(DISPLAY_SERVICE);
if (displayManager != null) {
   displayManager.registerDisplayListener(new DisplayManager.DisplayListener() {
      @Override
      public void onDisplayAdded(int displayId) {
      }

      @Override
      public void onDisplayChanged(int displayId) {
         synchronized (this) {
            mRenderer.onDisplayChanged();
         }
      }

      @Override
      public void onDisplayRemoved(int displayId) {
      }
   }, null);

   mGestureDetector = new GestureDetector(this,
         new GestureDetector.SimpleOnGestureListener() {
            @Override
            public boolean onSingleTapUp(MotionEvent event) {
               mCurrentX = event.getX();
               mCurrentY = event.getY();

               return true;
            }
            @Override
            public boolean onDoubleTap(MotionEvent event) {
               //mIsPut переходить false
               mCurrentX = event.getX();
               mCurrentY = event.getY();
               mIsPut = true;

               return true;
            }
            @Override
            public boolean onScroll(MotionEvent e1, MotionEvent e2,
                                    float distanceX, float distanceY) {

               //Можливість регулювати напрямок об’єкта в режимі реального часу
               if (mSelectedModel != -1) {
                  mRotateFactor -= (distanceX / 10);
                  mRotateFactor%=360;
                  if(mRotateFactor<0){
                     mRotateFactor += 360;
                  }
                  Matrix.rotateM(mModelMatrix, 0, -distanceX / 10, 0.0f, 1.0f, 0.0f);
                  String rotateFactor = String.format(Locale.getDefault(), " ", (int)mRotateFactor);
                  locate_rotate.setText(rotateFactor);
               }
               return true;
            }
         });
```

```
        mScaleDetector = new ScaleGestureDetector(this, new ScaleGestureDetector.SimpleOnScaleGestureListener()
{
                @Override
                public boolean onScale(ScaleGestureDetector detector) {
                    //Можливість зміни розміру в режимі реального часу
                    if (mSelectedModel != -1) {
                        mScaleFactor *= detector.getScaleFactor();
                        String scaleFactor = String.format(Locale.getDefault(), "  ", mScaleFactor*100 );
                        locate_scale.setText(scaleFactor);

                        Matrix.scaleM(mModelMatrix, 0,
                                detector.getScaleFactor(),
                                detector.getScaleFactor(),
                                detector.getScaleFactor());
                    }
                    return true;
                }
            });
    }

    mRenderer = new MainRenderer(this, new MainRenderer.RenderCallback(){
        @Override
        public void preRender() throws CameraNotAvailableException {
            if (mRenderer.isViewportChanged()) {
                Display display = getWindowManager().getDefaultDisplay();
                int displayRotation = display.getRotation();
                mRenderer.updateSession(mSession, displayRotation);
            }

            mSession.setCameraTextureName(mRenderer.getTextureId());

            Frame frame = mSession.update();
            if (frame.hasDisplayGeometryChanged()) {
                mRenderer.transformDisplayGeometry(frame);
            }

            PointCloud pointCloud = frame.acquirePointCloud();
            mRenderer.updatePointCloud(pointCloud);
            pointCloud.release();

            Collection<Plane> planes = mSession.getAllTrackables(Plane.class);
            for(Plane plane : planes){
                if(plane.getTrackingState() == TrackingState.TRACKING
                        && plane.getSubsumedBy() == null){
                    mRenderer.updatePlane(plane);
                }
            }

            final Camera camera = frame.getCamera();
            float[] projMatrix = new float[16];
            camera.getProjectionMatrix(projMatrix, 0, 0.1f, 100.0f);
            float[] viewMatrix = new float[16];
            camera.getViewMatrix(viewMatrix, 0);

            switch (mSelectedModel) {
                case TABLE:
                    if (!mModelInit[TABLE]) {
                        float position[] = calculateInitialPosition(mRenderer.getWidth(), mRenderer.getHeight(), projMatrix,
viewMatrix);

                        Matrix.setIdentityM(mModelMatrix, 0); //Створення матрці
                        Matrix.translateM(mModelMatrix, 0, position[0], position[1], position[2]);
                        Matrix.scaleM(mModelMatrix, 0, 0.02f, 0.02f, 0.02f); //Фіксує початковій розмір об’єкта
```

```
            mModelInit[TABLE] = true;
            mModelPut[TABLE] = false;
        }
        if (!mModelPut[TABLE]) {
            mRenderer.setTableModelMatrix(mModelMatrix);
        }
        mRenderer.setModelDraw(true, mModelPut[CHAIR], mModelPut[BED]);
        if (mModelInit[CHAIR] && !mModelPut[CHAIR]) {
            mModelInit[CHAIR] = false;
        }
        if (mModelInit[BED] && !mModelPut[BED]) {
            mModelInit[BED] = false;
        }
        break;
    case CHAIR:
        if (!mModelInit[CHAIR]) {
            float position[] = calculateInitialPosition(mRenderer.getWidth(),
                    mRenderer.getHeight(), projMatrix, viewMatrix);

            Matrix.setIdentityM(mModelMatrix, 0);
            Matrix.translateM(mModelMatrix, 0, position[0], position[1], position[2]);
            Matrix.scaleM(mModelMatrix, 0, 0.02f, 0.02f, 0.02f);

            mModelInit[CHAIR] = true;
            mModelPut[CHAIR] = false;
        }
        if (!mModelPut[CHAIR]) {
            mRenderer.setChairModelMatrix(mModelMatrix);
        }
        mRenderer.setModelDraw(mModelPut[TABLE], true, mModelPut[BED]);

        if (mModelInit[TABLE] && !mModelPut[TABLE]) {
            mModelInit[TABLE] = false;
        }
        if (mModelInit[BED] && !mModelPut[BED]) {
            mModelInit[BED] = false;
        }
        break;
    case BED:
        if (!mModelInit[BED]) {
            float position[] = calculateInitialPosition(mRenderer.getWidth(),
                    mRenderer.getHeight(), projMatrix, viewMatrix);

            Matrix.setIdentityM(mModelMatrix, 0);
            Matrix.translateM(mModelMatrix, 0, position[0], position[1], position[2]);
            Matrix.scaleM(mModelMatrix, 0, 0.02f, 0.02f, 0.02f);

            mModelInit[BED] = true;
            mModelPut[BED] = false;
        }
        if (!mModelPut[BED]) {
            mRenderer.setBedModelMatrix(mModelMatrix);
        }
        mRenderer.setModelDraw(mModelPut[TABLE], mModelPut[CHAIR], true);

        if (mModelInit[TABLE] && !mModelPut[TABLE]) {
            mModelInit[TABLE] = false;
        }
        if (mModelInit[CHAIR] && !mModelPut[CHAIR]) {
            mModelInit[CHAIR] = false;
        }
        break;
```

```
            default:
                break;
        }



        if (mIsPut) {
            List<HitResult> results = frame.hitTest(mCurrentX, mCurrentY);
            //HitTest для отримання результату
            for (HitResult result : results) {
                Trackable trackable = result.getTrackable();
                Pose pose = result.getHitPose();
                float[] modelMatrix = new float[16];
                pose.toMatrix(modelMatrix, 0);

                //Функція для задання напрямку та розміру
                Matrix.scaleM(modelMatrix, 0, mScaleFactor, mScaleFactor, mScaleFactor); //Збільшити/зменшити
матрицю
                Matrix.rotateM(modelMatrix, 0, mRotateFactor, 0.0f, 1.0f, 0.0f); //Поворот на співвідношення

                mScaleFactor = 0.02f;
                if (trackable instanceof Plane && ((Plane) trackable).isPoseInPolygon(result.getHitPose())) {
                    switch (mSelectedModel) {
                        case TABLE:
                            if (!mModelPut[TABLE]) {
                                mModelPut[TABLE] = true;

                                //Можливість запобігти обертанню та масштабуванню
                                mSelectedModel = -1;

                                System.arraycopy(modelMatrix, 0, mTableModelMatrix, 0, 16);
                                Matrix.setIdentityM(mModelMatrix, 0);

                                mIsPut = false;

                                runOnUiThread(new Runnable() {
                                    @Override
                                    public void run() {
                                        mTextView.setText(getString(R.string.table_put));
                                    }
                                });
                                TimerTask textTask = new TimerTask() {
                                    @Override
                                    public void run() {
                                        runOnUiThread(new Runnable() {
                                            @Override
                                            public void run() {
                                                mTextView.setText(getString(R.string.not_selected));
                                            }
                                        });
                                    }
                                };
                                Timer textTimer = new Timer();
                                textTimer.schedule(textTask, 2000);
                            }
                            break;
                        case CHAIR:
                            if (!mModelPut[CHAIR]) {
                                mModelPut[CHAIR] = true;
                                mSelectedModel = -1;
                                System.arraycopy(modelMatrix, 0, mChairModelMatrix, 0, 16);
                                Matrix.setIdentityM(mModelMatrix, 0);
                                mIsPut = false;
```

```
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        mTextView.setText(getString(R.string.chair_put));
                    }
                });
                TimerTask textTask = new TimerTask() {
                    @Override
                    public void run() {
                        runOnUiThread(new Runnable() {
                            @Override
                            public void run() {
                                mTextView.setText(getString(R.string.not_selected));
                            }
                        });
                    }
                };
                Timer textTimer = new Timer();
                textTimer.schedule(textTask, 2000);
            }
            break;
        case BED:
            if (!mModelPut[BED]) {
                mModelPut[BED] = true;
                mSelectedModel = -1;
                System.arraycopy(modelMatrix, 0, mBedModelMatrix, 0, 16);
                Matrix.setIdentityM(mModelMatrix, 0);
                mIsPut = false;
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        mTextView.setText(getString(R.string.bed_put));
                    }
                });
                TimerTask textTask = new TimerTask() {
                    @Override
                    public void run() {
                        runOnUiThread(new Runnable() {
                            @Override
                            public void run() {
                                mTextView.setText(getString(R.string.not_selected));
                            }
                        });
                    }
                };
                Timer textTimer = new Timer();
                textTimer.schedule(textTask, 2000);
            }
            break;
    }
  }
  if (!mIsPut) {
    break;
  }
}
if (mIsPut) {
  runOnUiThread(new Runnable() {
    @Override
    public void run() {
      mTextView.setText(getString(R.string.not_valid_position));
    }
  });
  TimerTask textTask = new TimerTask() {
```

```
                @Override
                public void run() {
                    runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                            switch (mSelectedModel) {
                                case TABLE:
                                    mTextView.setText(getString(R.string.table_selected));
                                    break;
                                case CHAIR:
                                    mTextView.setText(getString(R.string.chair_selected));
                                    break;
                                case BED:
                                    mTextView.setText(getString(R.string.bed_selected));
                                    break;
                                default:
                                    mTextView.setText(getString(R.string.not_selected));
                                    break;
                            }
                        }
                    });
                }
            };
            Timer textTimer = new Timer();
            textTimer.schedule(textTask, 2000);
        }
        mIsPut = false;
    }


    //Розміщення потрібного об’єкту на його місці
    if (mModelPut[TABLE]) {
        mRenderer.setTableModelMatrix(mTableModelMatrix);
        mRenderer.updateTableViewMatrix(viewMatrix);
        mModelInit[TABLE] = false;
    }
    if (mModelPut[CHAIR]) {
        mRenderer.setChairModelMatrix(mChairModelMatrix);
        mRenderer.updateChairViewMatrix(viewMatrix);
        mModelInit[CHAIR] = false;
    }
    if (mModelPut[BED]) {
        mRenderer.setBedModelMatrix(mBedModelMatrix);
        mRenderer.updateBedViewMatrix(viewMatrix);
        mModelInit[BED] = false;
    }

    mRenderer.setProjectionMatrix(projMatrix);
    mRenderer.updateViewMatrix(viewMatrix);
  }
});
mSurfaceView.setPreserveEGLContextOnPause(true);
mSurfaceView.setEGLContextClientVersion(2);
mSurfaceView.setEGLConfigChooser(8, 8, 8, 8, 16, 0);
mSurfaceView.setRenderer(mRenderer);


//save picutre
btn_capture_locate = (Button)findViewById(R.id.btn_capture_locate);
btn_capture_locate.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View v) {
    isSaveClick = true;
```

```
            mRenderer.printOptionEnable = isSaveClick;
            Toast.makeText(getApplicationContext(), "Saved!", Toast.LENGTH_SHORT).show();
        }
    });



}

@Override
public boolean onTouchEvent(MotionEvent event) {
    mGestureDetector.onTouchEvent(event);
    mScaleDetector.onTouchEvent(event);
    switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN:
            break;
        case MotionEvent.ACTION_UP:
            break;
    }
    return true;
}

@Override
protected void onPause() {
    super.onPause();

    mSurfaceView.onPause();
    mSession.pause();
}

@Override
protected void onResume() {
    super.onResume();


    try {
        if (mSession == null) {
            switch (ArCoreApk.getInstance().requestInstall(this, mUserRequestedInstall)) {
                case INSTALLED:
                    mSession = new Session(this);
                    Log.d(TAG, "ARCore Session created.");
                    break;
                case INSTALL_REQUESTED:
                    mUserRequestedInstall = false;
                    Log.d(TAG, "ARCore should be installed.");
                    break;
            }
        }
    }
    catch (UnsupportedOperationException e) {
        Log.e(TAG, e.getMessage());
    } catch (UnavailableApkTooOldException e) {
        e.printStackTrace();
    } catch (UnavailableDeviceNotCompatibleException e) {
        e.printStackTrace();
    } catch (UnavailableUserDeclinedInstallationException e) {
        e.printStackTrace();
    } catch (UnavailableArcoreNotInstalledException e) {
        e.printStackTrace();
    } catch (UnavailableSdkTooOldException e) {
        e.printStackTrace();
    }
```

```
    mConfig = new Config(mSession);
    if (!mSession.isSupported(mConfig)) {
        Log.d(TAG, "This device is not support ARCore.");
    }
    mSession.configure(mConfig);
    try {
        mSession.resume();
    } catch (CameraNotAvailableException e) {
        e.printStackTrace();
    }

    mSurfaceView.onResume();
    mSurfaceView.setRenderMode(GLSurfaceView.RENDERMODE_CONTINUOUSLY);
}


//Функція утримання місця, де буде створений об’єкт
public float[] calculateInitialPosition(int width, int height, float[] projMat, float[] viewMat) {
    return getScreenPoint(width / 2, height - 300, width, height, projMat, viewMat);
}

public float[] getScreenPoint(float x, float y, float w, float h,
                              float[] projMat, float[] viewMat) {
    float[] position = new float[3];
    float[] direction = new float[3];

    x = x * 2 / w - 1.0f;
    y = (h - y) * 2 / h - 1.0f;

    float[] viewProjMat = new float[16];
    Matrix.multiplyMM(viewProjMat, 0, projMat, 0, viewMat, 0);

    float[] invertedMat = new float[16];
    Matrix.setIdentityM(invertedMat, 0);
    Matrix.invertM(invertedMat, 0, viewProjMat, 0);

    float[] farScreenPoint = new float[]{x, y, 1.0F, 1.0F};
    float[] nearScreenPoint = new float[]{x, y, -1.0F, 1.0F};
    float[] nearPlanePoint = new float[4];
    float[] farPlanePoint = new float[4];

    Matrix.multiplyMV(nearPlanePoint, 0, invertedMat, 0, nearScreenPoint, 0);
    Matrix.multiplyMV(farPlanePoint, 0, invertedMat, 0, farScreenPoint, 0);

    position[0] = nearPlanePoint[0] / nearPlanePoint[3];
    position[1] = nearPlanePoint[1] / nearPlanePoint[3];
    position[2] = nearPlanePoint[2] / nearPlanePoint[3];

    direction[0] = farPlanePoint[0] / farPlanePoint[3] - position[0];
    direction[1] = farPlanePoint[1] / farPlanePoint[3] - position[1];
    direction[2] = farPlanePoint[2] / farPlanePoint[3] - position[2];

    normalize(direction);

    position[0] += (direction[0] * 0.1f);
    position[1] += (direction[1] * 0.1f);
    position[2] += (direction[2] * 0.1f);

    return position;
}

private void normalize(float[] v) {
    double norm = Math.sqrt(v[0] * v[0] + v[1] * v[1] + v[2] * v[2]);
```

```
        v[0] /= norm;
        v[1] /= norm;
        v[2] /= norm;
    }

    private void hideStatusBarAndTitleBar(){
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON,
                WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
    }

    public void onTableButtonClicked(View view) {
        mSelectedModel = TABLE;
        mTextView.setText(getString(R.string.table_selected));
    }

    public void onChairButtonClicked(View view) {
        mSelectedModel = CHAIR;
        mTextView.setText(getString(R.string.chair_selected));
    }

    public void onBedButtonClicked(View view) {
        mSelectedModel = BED;
        mTextView.setText(getString(R.string.bed_selected));
    }
}
```

MainActivity

```
package com.arcore.ruler;

import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.os.Environment;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AppCompatActivity;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import java.io.File;

public class MainActivity extends AppCompatActivity {

    public static final int IMAGE_GALLERY_REQUEST = 20;
    Button btn_measure;
    Button btn_locate;
    Button btn_gallery;
    Button btn_exit;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
        File dir_image = new File(Environment.getExternalStorageDirectory() + File.separator + "Ruler");
        dir_image.mkdirs();

        btn_measure = (Button)findViewById(R.id.btn_measure);
        btn_locate = (Button)findViewById(R.id.btn_locate);
        btn_gallery = (Button)findViewById(R.id.btn_gallery);
        btn_exit = (Button)findViewById(R.id.btn_exit);

        btn_measure.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(getApplicationContext(), MeasureActivity.class);
                startActivity(intent);
            }
        });

        btn_locate.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(getApplicationContext(), LocateActivity.class);
                startActivity(intent);
            }
        });
        btn_gallery.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                File[] imageFiles;
                imageFiles = new File(Environment.getExternalStorageDirectory().getAbsolutePath()+"/Ruler").listFiles();
                if(imageFiles.length<1){
                    Toast.makeText(getApplicationContext(),"Нет сохраненных изображений!",
Toast.LENGTH_SHORT).show();
                }
                else{
                    Intent intent = new Intent(getApplicationContext(), GalleryActivity.class);
                    startActivity(intent);
                }
            }
        });
        btn_exit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
            }
        });
    }

    public boolean onCreateOptionsMenu(Menu menu){
        super.onCreateOptionsMenu(menu);
        MenuInflater mainInflater = getMenuInflater();
        mainInflater.inflate(R.menu.main_option, menu);
        return true;
    }

    public boolean onOptionsItemSelected(MenuItem item){
        switch (item.getItemId()){
            case R.id.a:
                Toast.makeText(getApplicationContext(), "Яковлева Владислава", Toast.LENGTH_SHORT).show();
                return true;
            case R.id.b:
                Toast.makeText(getApplicationContext(), "Ruler v1.0", Toast.LENGTH_SHORT).show();
                return true;
            case R.id.c:
```

```
            Toast.makeText(getApplicationContext(), "Перейти к руководству", Toast.LENGTH_SHORT).show();
            Intent intent = new Intent(getApplicationContext(), ManualActivity.class);
            startActivity(intent);
            return true;
        }
        return false;
    }

    private void requestCameraPermission(){
        if(ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA)
                != PackageManager.PERMISSION_GRANTED){
            ActivityCompat.requestPermissions(this,
                    new String[]{Manifest.permission.CAMERA}, 0);
        }
    }

    private void requestMemoryPermission(){
        if(ContextCompat.checkSelfPermission(this, Manifest.permission.WRITE_EXTERNAL_STORAGE)
                != PackageManager.PERMISSION_GRANTED){
            ActivityCompat.requestPermissions(this,
                    new String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE}, 0);
        }
    }

    @Override
    protected void onResume() {
        super.onResume();
        requestCameraPermission();
        requestMemoryPermission();
    }

    /*public void onGalleryClicked(View v) {
        Intent photoPickerIntent = new Intent(Intent.ACTION_PICK);

        File pictureDirectory =
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES);
        String pictureDirectoryPath = pictureDirectory.getPath();

        Uri data = Uri.parse(pictureDirectoryPath);

        photoPickerIntent.setDataAndType(data, "image/*");

        startActivityForResult(photoPickerIntent, IMAGE_GALLERY_REQUEST);
    }*/
}
```

ManualActivity

```
package com.arcore.ruler;

import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.ImageView;

public class ManualActivity extends AppCompatActivity {

    private int num = 0;

    private int max = 17;

    int[] imgs = new int[max];
```

```
ImageView imageView, imageView2;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_manual);
    setTitle("Руководство");

    imageView = (ImageView)findViewById(R.id.changeImage);
    imageView2 = (ImageView)findViewById(R.id.changeImage2);

    for(int i = 0;i<max;i++)
    {
        imgs[i] = getApplicationContext().getResources().getIdentifier("img"+i, "drawable", getPackageName());
    }
}
public void onSkipButtonClick(View view){
    Intent intent = new Intent(getApplicationContext(), MainActivity.class);
    startActivity(intent);
}
public void onNextButtonClick(View view){
    num %= 17;

    imageView.setVisibility(View.GONE);
    imageView2.setVisibility(View.VISIBLE);

    switch (num)
    {
        case 0:
            imageView2.setImageResource(imgs[0]);
            break;
        case 1:
            imageView2.setImageResource(imgs[1]);
            break;
        case 2:
            imageView2.setImageResource(imgs[2]);
            break;
        case 3:
            imageView2.setImageResource(imgs[3]);
            break;
        case 4:
            imageView2.setImageResource(imgs[4]);
            break;
        case 5:
            imageView2.setImageResource(imgs[5]);
            break;
        case 6:
            imageView2.setImageResource(imgs[6]);
            break;
        case 7:
            imageView2.setImageResource(imgs[7]);
            break;
        case 8:
            imageView2.setImageResource(imgs[8]);
            break;
        case 9:
            imageView2.setImageResource(imgs[9]);
            break;
        case 10:
            imageView2.setImageResource(imgs[10]);
            break;
        case 11:
            imageView2.setImageResource(imgs[11]);
```

```
                break;
            case 12:
                imageView2.setImageResource(imgs[12]);
                break;
            case 13:
                imageView2.setImageResource(imgs[13]);
                break;
            case 14:
                imageView2.setImageResource(imgs[14]);
                break;
            case 15:
                imageView2.setImageResource(imgs[15]);
                break;
            case 16:
                imageView2.setImageResource(imgs[16]);
                break;
            default:
                break;
        }
        num++;
    }
}
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом_Яковлєва.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом_Яковлєва.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF.
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму.
Презентація	
Презентація_Яковлєва.ppt	Презентація кваліфікаційної роботи.

ВІДГУК

на кваліфікаційну роботу бакалавра
на тему:
"Розробка програмного забезпечення під керуванням ОС Android для виміру фізичних об’єктів за допомогою камери мобільного телефону"
студентки групи 121-11-1 Яковлєвої Владислави Олексіївни

В результаті виконання данної кваліфікаційної роботи було розроблене програмне забезпечення під керуванням ОС Android для виміру фізичних розмірів об’єктів за допомогою камери мобільного телефону.

Актуальність розробленого програмного продукту обумовлена зростанням популярності досліджень у напрямі доповненою реальності для мобільних додатків.

Розроблене програмне забезпечення може бути використане як основа для більш специфікованих додатків з використанням технології доповненої реальності в багатьох сферах. Для реалізації програмного забезпечення була використана мова програмування Java. В якості додаткового набору інструментів розробки була обрана бібліотека ARCore від компанії Google.

Працездатність представленої програми підтверджена налагоджувальними випробуваннями та тестуванням програми.

В економічному розділі визначено трудомісткість розробленої кваліфікаційної роботи проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Тема кваліфікаційної роботи безпосередньо пов'язана з об'єктом діяльності бакалавра за напрямом підготовки 121 «Інженерія програмного забезпечення».

Оформлення пояснювальної записки до кваліфікайційної роботи проекту виконано відповідно до стандартів на програмну документацію. Недоліком розглянутої роботи є неповний опис функціонування програмного забезпечення, невелика кількість пояснювальних коментарів і недостатній опис алгоритму реалізації програмного забезпечення.

Кваліфікаційна робота виконана самостійно та заслуговує оцінки 94 бала «Відмінно», а студентка Яковлєва Владислава Олексіївна присвоєння їй кваліфікації бакалавра за спеціальністю «фахівець з розробки та тестування програмного забезпечення».

Керівник дипломного проекту
асистент каф. ПЗКС, к.т.н. **С.Д. Приходченко**

РЕЦЕНЗІЯ
на кваліфікаційну роботу бакалавра
на тему:
"Розробка програмного забезпечення під керіванням ОС Android для виміру фізичних розмірів об'єктів за допомогою камери мобільного телефону"
студентки групи 121-17-1 Яковлєвої Владислави Олексіївни

Кваліфікаційна робота на тему «Розробка програмного забезпечення під керуванням ОС Android для виміру фізичних розмірів об'єктів за допомогою камери мобільного телефону» виконана в повному обсязі, відповідно до технічного завдання.

Мета кваліфікаційної роботи: розробка додатку та алгоритмів для відстеження оточення за допомогою камери мобільного телефону.

У пояснювальній записці розглянуто необхідність створення і сфера застосування розробленого програмного забезпечення, виконано постановку завдання, опис вхідних і вихідних даних, розроблено логічну структуру програми, розроблено інформаційне забезпечення системи, наведені загальні відомості про додаток та його функціональне призначення, зазначені використовувані технічні засоби, визначені джерела, використані при розробці.

Для реалізації інформаційної системи була використана мова програмування Java. В якості додаткового набору інструментів розробки була обрана бібліотека ARCore від компанії Google.

Вважаю завдання і зміст кваліфікаційної роботи відповідним для перевірки ступеня підготовленості Яковлєвої В. О. за напрямом 121 «Інженерія програмного забезпечення».

Список літератури, наведений в роботі, налічує більше 20 джерел, що свідчить про вміння автора працювати з літературою та іншими джерелами інформації. Якість оформлення кваліфікаційної роботи можна визнати задовільною, він супроводжується достатньою кількістю малюнків. В роботі присутні заключні висновки. Працездатність цього програмного забезпечення підтверджується експлуатаційними випробуваннями.

Рівень теоретичної та практичної підготовки автора, логіка і стиль викладу матеріалу в цілому відповідає кваліфікаційним вимогам.

Недоліком розглянутої роботи є неповний опис функціонування програмного забезпечення, невелика кількість пояснювальних коментарів.

Кваліфікаційна робота виконана самостійно та заслуговує оцінки «відмінно», а студентка Яковлєва Владислава Олексіївна присвоєння їй кваліфікації бакалавра за спеціальністю «фахівець з розробки та тестування програмного забезпечення».

Рецензент дипломного проекту
к.т.н., доцент каф. БІТ **О.В. Герасіна**