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

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 «Програмна інженерія»

на тему: «Програмне забезпечення для розширення експозиції музею на основі технології доповненої реальності»

Виконав (-ла):
студент (-ка) IV курсу, групи КП-52
Саленко Антон Володимирович _____

Керівник:
Доцент кафедри ПЗКС, к.т.н., доцент,
Сулема Є.С. _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н.,
Онай М.В. _____

Рецензент:
В.о завідувача кафедри ММСА ІПСА, к.т.н., доцент,
Тимощук О.Л. _____

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

Київ – 2019 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) – 6.050103 «Програмна інженерія» (Програмне забезпечення комп'ютерних систем)

ЗАТВЕРДЖУЮ

Науковий керівник кафедри

_____ І.А. Дичка

«___»_____2018 р.

ЗАВДАННЯ

на дипломний проект студенту

Саленку Антону Володимировичу

1. Тема проекту «Програмне забезпечення для розширення експозиції музею на основі технології доповненої реальності», керівник проекту Сулема Євгенія Станіславівна, затверджені наказом по університету від «___»_____ 2019 р. №_____

2. Термін подання студентом проекту «16» червня 2019 р.

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

4. Зміст пояснювальної записки:
   - аналіз технологій доповненої реальності та обгрунтування теми дипломного проекту;
   - розроблення алгоритмічного забезпечення додатку доповненої реальності;
   - розроблення програмного забезпечення додатку доповненої реальності;
   - аналіз отриманих результатів.

5. Перелік обов'язкового графічного матеріалу:
   - діаграма класів (креслення);

– діаграма прецедентів (креслення);
– архітектура додатку (плакат);
– UI-схема додатку (плакат).

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2018 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	Вивчення літератури за тематикою проекту	14.11.2018	
2.	Розроблення та узгодження технічного завдання	28.11.2018	
3.	Розроблення структури програмного забезпечення	15.12.2018	
4.	Підготовка матеріалів першого розділу дипломного проекту	30.12.2018	
5.	Розроблення дизайну програмного забезпечення	03.02.2019	
6.	Підготовка матеріалів другого розділу дипломного проекту	20.02.2019	
7.	Розроблення програмного забезпечення	10.03.2019	
8.	Тестування програмного забезпечення	17.03.2019	
9.	Підготовка матеріалів третього розділу дипломного проекту	30.03.2019	
10.	Підготовка матеріалів четвертого розділу дипломного проекту	11.04.2019	
11.	Підготовка графічної частини дипломного проекту	21.04.2019	
12.	Оформлення документації дипломного проекту	26.05.2019	

Студент А.В. Саленко

Керівник проекту Є.С. Сулема

АНОТАЦІЯ

Даний дипломний проект присвячений розробленню програмного забезпечення для розширення музейної експозиції шляхом додавання віртуальних експозицій.

Розроблене програмне забезпечення являє собою мобільний додаток, який надає змогу переглядати віртуальні експозиції у доповненій реальності за допомогою технології відстеження міток, завантажувати віртуальні експозиції з віддаленого сервера, робити фото та переглядати мапу музею.

У даному дипломному проекті було розроблено архітектуру, алгоритм орієнтації об’єкту за допомогою гіроскопу, модуль завантаження ресурсів з віддаленого сервера, зручний користувацький інтерфейс та проаналізовано залежність якості міток від типу зображень які для них використовуються.

ABSTRACT

This diploma project is devoted to the development of software for expanding museum exposition by adding virtual expositions.

Developed software is a mobile application that allows you to view virtual expositions in augmented reality using image tracking technology, download virtual expositions from a remote server, take photos and view the map of the museum.

In this diploma project developed an architecture, an algorithm for orienting an object with a gyroscope, a module for downloading resources from a remote server, a user-friendly interface and analyzed the quality dependence of markers on the type of images used for them.

ДП.045480-01-90 Програмне забезпечення для розширення експозиції музею на основі технології доповненої реальності. Відомість проекту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проекту		
ДП.045480-02-91	Програмне забезпечення	4	
	для розширення		
	експозиції музею на		
	основі технології		
	доповненої реальності.		
	Технічне завдання		
ДП.045480-03-81	Програмне забезпечення	53	
	для розширення		
	експозиції музею на		
	основі технології		
	доповненої реальності.		
	Пояснювальна записка		
ДП.045480-04-51	Програмне забезпечення	4	
	для розширення		
	експозиції музею на		
	основі технології		
	доповненої реальності.		
	Програма та методика		
	тестування		
ДП.045480-05-34	Програмне забезпечення	6	
	для розширення		
	експозиції музею на		
	основі технології		

	доповненої реальності.		
	Керівництво користувача		
ДП.045480-06-99	Програмне забезпечення	1	
	для розширення		
	експозиції музею на		
	основі технології		
	доповненої реальності.		
	Функціональні		
	можливості програмного		
	забезпечення. UML		
	діаграма прецедентів.		
ДП.045480-07-99	Програмне забезпечення	1	
	для розширення		
	експозиції музею на		
	основі технології		
	доповненої реальності.		
	Архітектура ПЗ.		
	Діаграма класів.		
ДП.045480-08-98	Програмне забезпечення	1	
	для розширення		
	експозиції музею на		
	основі технології		
	доповненої реальності.		
	Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2018 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗШИРЕННЯ ЕКСПОЗИЦІЇ МУЗЕЮ НА ОСНОВІ ТЕХНОЛОГІЇ ДОПОВНЕНОЇ РЕАЛЬНОСТІ

Технічне завдання

ДП.045480-02-91

"ПОГОДЖЕНО"

Керівник проекту:

_____ Є.С. Сулема

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ А.В. Саленко

2018

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Програмне забезпечення для розширення експозиції музею на основі технології доповненої реальності.

Галузь застосування: мультимедійні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проектування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для розширення експозиції музею віртуальними експозиціями на основі доповненої реальності.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Програмне забезпечення повинно забезпечувати такі функціональні можливості:

1) можливість розпізнавання маркерів;
2) можливість завантажити віртуальні експозиції з віддаленого серверу;
3) можливість переглядати інтерактивну мапу;
4) можливість перегляду віртуальних експозицій у доповненій реальності;
5) імітація доповненої реальності за недостачі ресурсів апаратного забезпечення;
6) можливість зробити фотознімок та поділитися ним у соціальних мережах;

7) автоматичне фокусування камери.

Розробку виконано з використанням мови C# і графічного рушія Unity Engine для реалізації технології доповненої реальності використано Vuforia Engine.

Додаткові вимоги:

1) аналітика переглядів віртуальних експозицій;

2) зміна мови інтерфейсу користувача.

5. ВИМОГИ ДО ПРОЕКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проекту повинна бути розроблена наступна документація:

1) пояснювальна записка;

2) програма та методика тестування;

3) керівництво користувача;

4) креслення:

– «Архітектура ПЗ. Діаграма класів»;

– «Функціональні можливості програмних засобів. UML діаграма прецедентів».

6. ЕТАПИ ПРОЕКТУВАННЯ

Вивчення літератури за тематикою проекту…….. 14.11.2018

Розроблення та узгодження технічного завдання……. 28.11.2018

Розроблення структури програмного забезпечення….. 15.12.2018

Розроблення дизайну інтерфейсу…… 03.02.2019

Програмна реалізація програмного забезпечення……. 10.03.2019

Тестування програмного забезпечення…….. 17.03.2019

Підготовка матеріалів графічної частини проекту…… 21.04.2019

Оформлення технічної документації проекту……... 26.05.2019

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗШИРЕННЯ ЕКСПОЗИЦІЇ МУЗЕЮ НА ОСНОВІ ТЕХНОЛОГІЇ ДОПОВНЕНОЇ РЕАЛЬНОСТІ

Пояснювальна записка

ДП.045480-03-81

"ПОГОДЖЕНО"

Керівник проекту:

_____ Є.С. Сулема

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ А.В. Саленко

2019

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

AR − Augmented Reality, або доповнена реальність

Маркер − фізичний об’єкт, інформація (зображення) якого використовується у плагіні Vuforia для відстежування його положення відносно камери

ПЗ − програмне забезпечення

UI − інтерфейс користувача

ARKit, ARCore, Vuforia − найпопулярніші на ринку ПЗ для роботи із доповненою реальністю

JPG, PNG − поширені формати зображень

OpenGL, Metal − інтерфейси для роботи з графікою

SDK − набір засобів розробки

Unity Asset Store − онлайн магазин ресурсів для розробки на Unity

SVN − централізована система управління версіями

GIT − розподілена система управління версіями

World Tracking − технологія орієнтації у просторі

CG − мова програмування шейдерів

Шейдер − програма для обробки графіки за допомогою GPU

DataMatrix − матричний штриховий код

GPU − графічний процесор

HTTP − протокол передачі гіпертексту

World Map − технологія збереження відносних позицій відсканованих об’єктів

C# − мова програмування

Unity Engine − кросплатформений графічний рушій

Unreal Engine − кросплатформений графічний рушій

COP − архітектурний шаблон

MVCS − архітектурний шаблон

Android − найпопулярніша платформа для мобільних девайсів

iOS − платформа для мобільних девайсів

Гіроскоп − механо-електронний пристрій для визначення орієнтації у просторі

FPS – кількість кадрів в секунду

ВСТУП

Основна мета дипломної роботи – розроблення програмного забезпечення доповненої реальності для розширення експозиції музею. Додаток дозволяє відстежувати маркери та відтворювати на них анімаційний контент, який несе в собі візуальну та аудіальну інформацію про експонат чи подію в історії. Підставою для розроблення додатку стала проблема зниження популярності музеїв серед молодого населення.

В наш час багато підлітків не зацікавлені відвідуванням музею, а шкільні екскурсії у музей для більшості учнів перетворюються у нудний процес блукання по музею в пошуках сидіння. Тому музеї потребують інноваційного підходу для вирішення цієї проблеми. Одним із таких підходів є інтеграція технології доповненої реальності який повною мірою почав себе проявляти всередині 2017 року з появою технологій ARKit та ARCore які вивели технологію у маркетинг.

Програмне забезпечення розроблене в цій дипломній роботі надає такі можливості:

1) відстеження положення маркеру відносно камери телефону та програвання на ньому анімаційного контенту з освітньою функцією;
2) перегляд інтерактивної мапи;
3) взаємодія з інтерактивним анімаційним контентом;
4) динамічне оновлення анімаційного контенту на сховищі;
5) зручний UI;
6) використання в ролі зручного аудіо-гіда.

Якщо скласти всі вищезгадані компоненти в одне ціле, то отримаємо зручний інноваційний продукт який виконує роль інтерактивного гіда та зацікавить молоде покоління.

Не зважаючи на те, що даний тип програми не є єдиним рішенням на ринку ПЗ, в цьому дипломному проекті буде проаналізовано вже існуючі рішення і розроблено останню версію продукту з урахуванням всіх мінусів існуючих розробок.

Роззроблене ПЗ не потребує від користувача спеціальних знань у технології доповненої реальності та навичок програмування, що робить його доступним для кожного.

1. АНАЛІЗ ТЕХНОЛОГІЙ ДОПОВНЕНОЇ РЕАЛЬНОСТІ ТА ОБГРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ

1.1. Обґрунтування актуальності проблеми

Основна проблема яку вирішує додаток – низька популярність музеїв серед молодого населення. Проблема закладена у асоціації відвідування музеїв з нудним проведенням часу. В наш час відвідування музею не дає відвідувачу всієї тієї інформації яку може надати, тому що більшість відвідувачів просто переглядають експонати не цікавлячись детальною інформацією про нього чи про події з якими він зв’язаний, а аудіогід вартує додаткової оплати. Тому використання інноваційного підходу у вигляді технології доповненої реальності – хороше рішення підняти популярність музеїв.

Загалом доповнена реальність – це технологія яка доповнює реальність віртуальними об’єктами. Існують різні способи реалізації цієї технології, наприклад технологія комп’ютерного бачення (Vuforia) яка дає можливість знаходити та відслідковувати рух заданого об’єкту. В доповнення до цього використовується відслідковування переміщення пристрою за допомогою акселерометру та гіроскопу, що використовується в технологіях ARKit та ARCore для можливості синхронізації координат реального простору з віртуальними. Вперше ця технологія з’явилася у 1992 році, коли Луї Розенберг створив першу систему AR в дослідницькій лабораторії ВПС США [4]. Зараз ця технологія набула високої популярності і дійшла навіть до маркетингу та реклами різноманітних продуктів. За приклад можна взяти компанію Coca-Cola та їх мобільний додаток який відтворює анімацію на пляшці напою в доповненій реальності.

Основні переваги використання доповненої реальності в маркетингу:

1) інноваційність – технологія з’явилася відносно недавно і приваблює коричстувачів;
2) розважливий характер;
3) багато прикладів успішних маркетингових рішень заснованих на цій технології;
4) можливість використання смартфонами – мобільність та зручність.

Музею нічим себе зацікавити новим відвідувачам окрім як інноваціями та відомими на весь світ експонатами. Так як відомі на весь світ експонати зазвичай недоступні для менш відомих музеїв то інновація підходить більше для приваблення нових відвідувачів. Окрім проблеми низької популярності, існує ще проблема незручних гідів. Тайванські дослідники дослідили ефективність використання доповненої реальності, спостерігаючи за поведінкою музейних груп з екскурсоводом, аудіогідом и AR-додатком. Дослідження показало, що після екскурсії найбільшу інформованість і захопленість проявила саме та група, яка використовувала мобільний гід з доповненою реальністю.

Дослідження проблем виявило основні недоліки які спричинили низьку популярність музеїв та можливі варіанти їх вирішення.

1.2. Порівняльний аналіз технологій доповненої реальності

Проаналізувавши різні джерела інформації а також продукти що зв’язані з темою доповненої реальності для мобільних девайсів було виділено три основні технології: ARKit, ARCore та Vuforia.

1.2.1. *Технологія ARKit*

ARKit – технологія доповненої реальності що використовує інформацію про орієнтацію девайсу у просторі, його зміщення та зображення з камери для того щоб прикріпити якорі до точок простору і відносно них визначати положення девайсу у майбутньому. Якір – це

зв’язок координат певної точки на зображенні (кількох точок) та віртуальними координатами, що виділяються серед інших і які можна буде знайти після руху девайса. За допомогою такого якоря можна прив’язати віртуальний об’єкт, наприклад 3D модель до координат простору реального світу. Ця технологія називається World Tracking. Її основна мета — побудувати віртуальну мапу реального світу яка складається з якорів. Для того щоб визначити координати якоря відносно телефону використовується технологія Orientation Tracking що використовує гіроскоп для визначення орієнтації девайсу у просторі за трьома осями його повороту та акселерометр для визначення прискорення. Ці дані, а саме орієнтація, прискорення та набір зображень обробляються у такому порядку, спочатку за допомогою визначеної орієнтації і прискорення визначаються точки координат простору відносно початкового положення девайсу. У цей же час обробляються зображення і визначається напрям у якому лежать різні об’єкти які можна розпізнати на зображеннях. Після того як девайс змінив своє положення то напрям у якому знаходяться різні об’єкти змінився на різну величину. Із цих змін за допомогою алгоритму визначаються координати об’єктів відносно девайсу. Чим більше користувач рухає девайс тим більше унікальних даних надходить і World Map стає більш чіткою. Але є кілька обмежень, наприклад якщо просто навести камеру на стіну то нічого особливого відстежуватись не буде, а якщо дуже різко рухати девайс то дані які надходять на обробку починають втрачати зв’язок між собою, що спричиняє сильне відхилення відстежених координат від реальних.

Відстеження горизонтальних поверхонь використовує World Map щоб визначити поверхні на які можна виставити віртуальні об’єкти.

Технологія ARKit розроблена компанією Apple та працює лише на платформі iOS тому для кросплатформного додатку потрібно використовувати ще одну із нижче наведених технологій, що працюватиме на платформі Android, так як частка ринку продукції для мобільних девайсів яка підтримує цю технологію близько 20% [14].

Мінімальні вимоги технології:

1) iOS 12;
2) iPhone 6s та новіші, список можна переглянути на офіційному сайті Apple.

1.2.2. *Технологія ARCore*

ARCore – технологія доповненої реальності розроблена компанією Google що працює на платформах Android, ринок якої серед мобільних девайсів близько 80% та iOS (використовуючи технологію ARKit). Ця технологія використовує ті ж технології для досягнення цілі що й технологія ARKit в плані відстеження горизонтальних поверхонь. Але ця технологія дозволяє відстежувати рівні поверхні нахилені під будь-яким кутом [16].

Мінімальні вимоги технології:

1) Android 7.0+ або iOS аналогічно до технології ARKit;
2) список підтримуваних пристроїв можна переглянути на офіційному сайті Google Developers.

1.2.3. *Технологія Vuforia*

Vuforia – кросплатформна технологія що використовує технології ARKit та ARCore для відстеження поверхонь та свою власну технологію відстеження маркерів. Маркер це наперед оброблене зображення алгоритмом пошуку найбільш виразних точок, яке в подальшому відслідковується в реальному світі за допомогою оброблених алгоритмом даних [13].

Алгоритм пошуку маркера складається з кількох етапів:

1. Обробка зображення за допомогою гістограми орієнтованих градієнтів. Групування сусідніх орієнтованих градієнтів та пошук груп градієнти і напрям яких співпадають. В результаті це можна оцінити як пошук місць на зображенні де є різкі переходи кольору та перетини ліній.

2. Порівняння орієнтованих градієнтів зображення з камери та оригінального зображення, яке було оброблене наперед.
3. Визначення границь знайденого зображення та його орієнтації у просторі.

Після того як маркер знайдено відбувається його відстеження на що витрачається менше ресурсів а також для відстеження відносна оцінка подібності знайденого зображення до реального нижча ніж для його пошуку.

Мінімальні вимоги технології:

1) Android 4.1+, iOS 9+;
2) двоядерний процесор;
3) 1 Gb оперативної пам’яті;
4) підтримка графічних інтерфейсів OpenGL ES 2.0, OpenGL ES 3.x, Metal.

1.3. Порівняльний аналіз графічних рушіїв

Проаналізувавши різні джерела інформації було визначено два найпопулярніші кросплатформна графічні рушії, з простою інтеграцією в них вище наведених технологій доповненої реальності.

1.3.1. *Unity3D Engine*

Unity Engine – простий в розумінні, налаштуванні кросплатформний графічний рушій для розробки дво- та тривимірних додатків та ігор, що працює на операційних системах Windows і OS X. Має широкі можливості в роботі з графікою, фізикою а також має інтегрований Vuforia SDK. За потреби на Unity Asset Store можна завантажити такі ж SDK для роботи з технологіями ARKit та ARCore. Створені за допомогою Unity застосування працюють під системами Windows, OS X, Android, Apple IOS, Linux, а також на гральних консолях Wii, PlayStation 3, Xbox 360.

Є можливість створювати інтернет-додатки за допомогою спеціального під’єднуваного модуля для браузера Unity що працює з

WebGL, а також за допомогою експериментальної реалізації в межах модуля Adobe Flash Player [11].

Технічні характеристики:

1) можливість писати скрипти на C#, JS;
2) ігровий рушій повністю пов'язаний із середовищем розробки. Це дозволяє випробовувати гру чи додаток прямо в редакторі;
3) зв’язувати ресурси можлива за допомогою Drag&Drop;
4) система наслідування класів чудово працює в редакторі;
5) система компонентно-орієнтованого програмування COP що використовується в рушії є зручною для роботи з скриптами які відповідають за поведінку об’єкта;
6) підтримка майже усіх розширень файлів моделей, текстур, аудіо, відео та інших ресурсів;
7) вбудований багатофункціональний генератор ландшафту;
8) вбудована обгортка для роботи з Http-запитами;
9) існує рішення для спільної розробки — Asset Server. Також можна використовувати зручний для користувача спосіб контролю версій. Наприклад, SVN або Source Gear;
10) фізика реалізована у вигляді відомих фізичних рушіїв Physics 2D та Physics;
11) підтримка графічних інтерфейсів OpenGL, OpenGL ES, DirectX, Metal, Vulkan.

Сервер ресурсів Unity це хороше рішення для контролю версій для всіх ресурсів, як скриптів так і моделей, текстур, аудіофайлів та інших. Він подібний до системи контролю версій Git і простий у використанні, а також має ряд переваг, такі як простий злиття деяких генерованих Unity ресурсів. Великі проекти з тисячами мегабайтних файлів ресурсів піддаються легкому керуванню, та одночасній роботі над ними кількома розробниками. Налаштування імпорту ресурсів та інші метадані зберігаються разом з історією їх версій. Якщо файли змінюються, їх статус негайно оновлюється.

Перейменування і переміщення ресурсів не створює будь-яких перешкод для безперервного робочого процесу.

Сервер ресурсів Unity управляється базою даних PostgreSQL. PostgreSQL відомий своєю надійністю, цілісністю даних і легкістю адміністрування і відмінно справляється з робочим навантаженням гігантських проектів.

Цей ігровий двигун вкрай популярний серед інді-розробників та стартаперів. Однією з причин популярності є те, що в одному середовищі можна створити додаток, який буде запускатися майже на всіх пристроях в тому числі і на браузері.

Важливим показником є швидкість роботи. Універсальність і кросплатформність часто несе в собі збільшення навантаження на обчислювальні системи комп'ютера. На Unity розроблено безліч ігор, в тому числі, які працюють на консолях, але за рівнем графіки вони майже завжди поступалися топовим іграм періоду, в який виходили. Звичайно, на цьому двигуні можна створювати ігри з дивовижною графікою, але для цього потрібно вивчити досконало цю технологію [11].

Універсальність несе з собою складність пристосування до широких мас населення. Зазвичай на Unity створюють специфічні системи, а не системи для широкого використання. Це означає, що у цього двигуна є досить близька межа можливостей.

1.3.2. *Unreal Engine*

Unreal Engine – ігровий рушій, що розробляється і підтримується компанією Epic Games.

Перша гра, створена на цьому рушії – Unreal – з'явилася в 1998 році. З тих пір різні версії рушія були використані в більш ніж сотні ігор і інших проектів.

Написаний на мові C++, рушій дозволяє створювати ігри для більшості операційних систем і платформ: Microsoft Windows, Linux, Mac

OS і Mac OS X; консолей Xbox, Xbox 360, Xbox One, PlayStation 2, PlayStation 3, PlayStation 4, PSP, PS Vita, Wii, Dreamcast, GameCube і ін., а також на різних портативних пристроях, наприклад, пристроях Apple (iPad, iPhone), керованих системою iOS і інших. (Вперше робота з iOS була представлена в 2009 році, в 2010 році продемонстровано роботу рушію на пристрої з системою webOS) [12].

Для спрощення портування движок використовує модульну систему залежних компонентів; підтримує різні системи рендеринга (Direct3D, OpenGL, Pixomatic; в ранніх версіях: Glide, S3, PowerVR), відтворення звуку (EAX, OpenAL, DirectSound3D; раніше: A3D), засоби голосового відтворення тексту, розпізнавання мови, модулі для роботи з мережею та підтримуваних пристроїв введення.

1.3.3. *Результат порівняння*

Unity3D – рушій який частіше використовується для розробки малих додатків, ігор, які не потребують надсучасних технологій рендерингу та забезпечує швидку розробку програм завдяки меншим вимогам до розміру коду а також більшість проектів розроблених на цьому рушії орієнтовані на платформи Android та iOS. З іншого боку Unreal Engine використовується в основному для розробки ігор для ПК, ігрових консолей PlayStation та Xbox, в основному через можливість глибокого оптимізування коду та ресурсів які використовує програма. А також з допомогою нових технологій рендерингу картинка створена за допомогою Unreal Engine перевершує картинку Unity3D Engine. Так як проект орієнтований на платформи iOS та Android та мобільні девайси не мають можливості відкрити потенціал рендерінгу Unreal Engine то розробка буде вестись на графічному рушії Unity3D Engine.

1.4. Постановка задачі

В результаті проведеного дослідження можна сформулювати такі вимоги до програмного забезпечення віртуального музею на основі технології доповненої реальності:

1) ПЗ має мати алгоритм розпізнавання маркерів, так як це основна вимога для можливості використання доповненої реальності;
2) для ПЗ мають бути створені ідеальні, за оцінкою плагіну Vuforia, маркери для швидкого та безпомилкового трекінгу;
3) максимально простий і зручний UI, так як у користувачів не буде багато часу на його ознайомлення;
4) малий розмір додатку (<100 Мб) для можливості завантаження його із сервісів Google Play і App Store без використання Wi-Fi;
5) сервер або сервіс для зберігання анімаційного контенту поза межами додатку, для зменшення його початкового розміру та можливості керування анімаційним контентом;
6) модуль завантаження даних із віддаленого хранилища;
7) окремий додаток для перевірки якості маркерів;
8) модуль пакування та розпакування сторонніх ресурсів ПЗ для можливості викладення їх на віддалене хранилище;
9) набір анімаційного контенту, яким є озвучена 3D анімація з ефектами та можливістю інтеракції з персонажами;
10) модуль системи освітлення, який включає в себе перелік модифікованих surface шейдерів;
11) архітектурний модуль, який стане основою зв’язку інших модулів та забезпечить масштабування ПЗ;
12) модуль управління камерою для визначення типу фокусування та деяких інших функцій;
13) інтерактивна мапа музею для зручності користувачів.

1.5. Висновки до розділу 1

Згідно інформації, наведеної в розділі 1 проблема залишається актуальною і має підстави для вирішення. Рішенням цієї проблеми буде мобільний додаток на основі технології доповненої реальності що допоможе іншим та цікавим шляхом піднести відвідування музеїв. Для розробки цього додатку найкраще буде використати популярний сьогодні ігровий рушій Unity та також популярну технологію доповненої реальності базовану на маркерах Vuforia. Додаток вирішено зробити кросплатформним та мобільним для платформ Android та iOS.

2. РОЗРОБЛЕННЯ АЛГОРИТМІЧНОГО ЗАБЕЗПЕЧЕННЯ ДОДАТКУ ДОПОВНЕНОЇ РЕАЛЬНОСТІ

2.1. Алгоритм орієнтації об’єкту за допомогою гіроскопу

Гіроскоп – це пристрій що здатний реагувати на зміну орієнтації відносно його основи, якою виступає в нашому випадку мобільний телефон. В мобільному телефоні використовуються не звичайні механічні гіроскопи, а мікро-електромеханічні, що перетворюють кутову швидкість в електро-сигнал. Зараз вони досі недостатньо точні для проведення математичних обчислень та іноді виходять з ладу і розкид поточного значення збільшується до 20 градусів. Але у новіших моделях такі неполадки трапляються значно рідше, тому працювати з ними стало простіше [10].

Мета розробки даного алгоритму – створити імітацію доповненої реальності у разі втрати маркеру з поля зору сканера. Це є важливим пунктом у розробленні даного ПЗ тому, що користувач не завжди матиме змогу переглядати контент на маркері так як маркер лише один, а кількість потенційних користувачів які одночасно переглядають один анімаційний контент може бути великою.

Подібний алгоритм використовується у технологіях ARKit та ARCore для визначення орієнтації девайсу та орієнтуванню камери в сцені відносно визначеної орієнтації гіроскопу. Але у випадку ARKit поворот не потрібно інвертувати, що спрощує реалізацію. Але у даному ПЗ орієнтація камери заблокована, тому що нею керує один із скриптів технології Vuforia, і будь-яка зміна орієнтації камери може призвести до неочікуваних результатів.

В даному алгоритмі гіроскоп буде використовуватися для визначення орієнтації мобільного телефону. Орієнтація буде розкладатися на поворот відносно трьох осей координат. А ці повороти будуть орієнтувати у просторі анімаційний контент так щоб напрям вверх завжди співпадав з реальним напрямом вверх.

У лістигну 2.1 наведено реалізацію алгоритму мовою C#.

Лістинг 2.1. Алгоритм орієнтації за допомогою гіроскопу

```
if (SystemInfo.supportsGyroscope)
{
    var gyroRotation = Input.gyro.attitude;
    transform.LookAt(targetTransform);
    var rotation = Quaternion.Euler(-90f, 0f, 0f) *
        gyroRotation;
    transform.localRotation = Quaternion.Euler(0f, 180f, 0f) *
        Quaternion.Euler(0f, 0f, rotation.eulerAngles.z) *
        Quaternion.Euler(-rotation.eulerAngles.x, 0f, 0f) *
        transform.localRotation;
}
else
{
    transform.LookAt(targetTransform, targetTransform.up);
    transform.localRotation *= Quaternion.Euler(-35f, 180f, 0f);
}
```

Поетапно розглянемо реалізацію. Для початку визначаємо чи девайс підтримує гіроскоп. Якщо так тоді отримуємо його орієнтацію та повертаємо на —90 градусів по осі *X* тому що нулева ротація це коли телефон напрямлений камерою вниз, а нам потрібно щоб нулева ротація була тоді коли камера телефону напрямлена вперед. Опісля потрібно повернути об’єкт на відповідний поворот гіроскопа по осі *Z*, після цього на від’ємне значення повороту гіроскопа по осі *X* і застосувати їх до попереднього повороту об’єкту в напрямку камери. В тому випадку якщо гіроскоп не підтримується на девайсі то просто повертаємо об’єкт в напрямку камери і розвертаємо його вниз на 35 градусів так щоб ми могли дивитись на нього зверху. Під таким кутом зручно спостерігати за тим що відбувається на сцені. Ці обчислення проводяться кожного кадру за умови що маркер втрачено. Цей алгоритм допоможе виділитися серед конкурентів за допомогою імітації доповненої реальності, чого немає у подібних додатках на основі технології доповненої реальності.

2.2. Алгоритм масштабування мапи жестами

Для зручності перегляду мапи додано функцію її масштабування за допомогою жестів тачскріна. Для того щоб керування масштабуванням було зручним використовується принцип масштабування за допомогою розведення і зведення двох пальців. Основна властивість даного алгоритму це те що масштабований об’єкт збільшується прямопропорційно відношенню відстані між пальцями на початку жесту і в кінці.

У лістингу 2.2 наведено реалізацію алгоритму мовою C#.

Лістинг 2.2. Алгоритм масштабування

```
private void OnSwipe(IEvent eventData)
{
    Vector2 fingerPos;
    var swipe = eventData.data as SwipeModel;
    if (swipe.state == MainContextInput.stateTouch.STSwipeStart)
    {
        RectTransformUtility.
            ScreenPointToLocalPointInRectangle(mapHolder,
            swipe.point1, null, out fingerPos);
        swipeStartPos = fingerPos;
        startAnchorPos = mapTransform.anchoredPosition;
        return;
    }
    RectTransformUtility.
        creenPointToLocalPointInRectangle(mapHolder,
        swipe.point1, null, out fingerPos);
    var deltaPos = fingerPos - swipeStartPos;
    mapTransform.anchoredPosition = startAnchorPos +
        (Vector2)deltaPos;
    var size = mapTransform.sizeDelta / 2f;
    mapTransform.anchoredPosition = new Vector2(
        Mathf.Clamp(mapTransform.anchoredPos ition.x, -size.x,
            size.x),
        Mathf.Clamp(mapTransform.anchoredPosition.y, -size.y,
            size.y));
}

private void OnZoom(IEvent eventData)
{
    var zoom = eventData.data as ZoomModel;
    Vector2 fingerPos;
    if (zoom.state == MainContextInput.stateTouch.STZoomStart)
    {
RectTransformUtility.ScreenPointToLocalPointInRectangle(mapHolder,
(zoom.point1 + zoom.point2) / 2f, null, out fingerPos);
        swipeStartPos = fingerPos;
        startAnchorPos = mapTransform.anchoredPosition;
        zoomStartDistance = Vector3.Distance(zoom.point1,
zoom.point2);
        zoomStartScale = mapTransform.localScale.x;
        return;
    }
```

```
    Vector2 zoomedStartPos = swipeStartPos *
(Vector3.Distance(zoom.point1, zoom.point2) / zoomStartDistance);
    mapTransform.localScale = Vector3.one *
Mathf.Clamp(zoomStartScale * (Vector3.Distance(zoom.point1,
zoom.point2) / zoomStartDistance), 0.5f, 2f);

RectTransformUtility.ScreenPointToLocalPointInRectangle(mapHolder,
(zoom.point1 + zoom.point2) / 2f, null, out fingerPos);
    var deltaPos = fingerPos - zoomedStartPos;
    mapTransform.anchoredPosition = startAnchorPos +
(Vector2)deltaPos;
    var size = mapTransform.sizeDelta / 2f;
    mapTransform.anchoredPosition = new
Vector2(Mathf.Clamp(mapTransform.anchoredPosition.x, -size.x,
size.x), Mathf.Clamp(mapTransform.anchoredPosition.y, -size.y,
size.y));
}
```

Для розрахунку поточного розміру об’єкта використовується відношення відстані між поточними точками дотику та відстані між точками дотику на початку жесту. Позиція об’єкту розраховується за допомогою значення зміщення центру між двома точками дотику. Для цього зберігається позиція об’єкту на початку жесту, та позиція точки що знаходиться по центру між двома точками дотику. Для того щоб розрахувати позицію об’єкту переводимо їх у локальну систему координат об’єкту та обраховуємо зміщення. Початкову позицію домножуємо на відношення розміру об’єкту на початку жесту і на поточному етапі. Додаємо до обрахованої початкової позиції величину що дорівнює зміщенню центру між двома точками дотику в локальній системі координат. В кінці обмежуємо розмір об’єкту зверху значенням вдвічі більшим за стандартний і знизу вдвічі меншим за стандартний та позицію об’єкту його розмірами для того щоб не втратити об’єкт з поля зору. такий метод дозволить переглядати мапу зручно для користувача, тому що метод подібний до стандартного масштабування у більшості програм для перегляду фото.

2.3. Алгоритм розмиття заднього фону

Цей алгоритм подібний до алгоритму розмиття Гауса, але швидший і видає зображення меншої якості. Алгоритм розмиття Гауса не підійде для реалізації розмиття заднього фону, так як використовує багато ресурсів, а

саме для обчислення одного пікселя на вхід необхідно отримати велику кількість, 49 сусідніх пікселів для того щоб перемножити їх поелементно на матрицю розміром 7*7, що не є доцільним для використання на мобільних платформах так як сильно зростає час обробки кадру. Розроблений алгоритм використовує менше ресурсів, а також має можливість плавно змінювати силу розмиття. Алгоритм має високу швидкодію порівняно з алгоритмами розмиття за Гаусом так як обчислення дійсних коефіцієнтів Гаусової піраміди, а саме обчислення дійсних факторіалів, займає дуже багато часу. Для мобільних девайсів дуже важлива швидкість роботи і продуктивність, а подібні методи наведені вище не являються швидкими, тому не можуть бути застосованими. У лістингу 2.3 наведено один прохід шейдера, написаного мовою CG [7].

Лістинг 2.3. Blur Shader

```
Shader "Unlit/BlurShader"
{
Properties
{
	_MainTex("Texture", 2D) = "white" {}
	_Power("Power", Range(1, 2)) = 1
	_Offset ("Offset", Range(1, 32)) = 1
}
SubShader
{
	Cull Off ZWrite Off ZTest Always
	Pass
	{
		CGPROGRAM
		#pragma vertex vert
		#pragma fragment frag
		#include "UnityCG.cginc"
		struct v2f
		{
			float2 uv : TEXCOORD0;
			float4 vertex : SV_POSITION;
		};
		struct appdata
		{
			float4 vertex : POSITION;
			float2 uv : TEXCOORD0;
		};
		sampler2D _MainTex;
		float4 _MainTex_ST;
		float2 _MainTex_TexelSize;
		fixed _Offset;
		fixed _Power;
		v2f vert(appdata v)
```

```
            {
                  v2f o;
                  o.vertex = UnityObjectToClipPos(v.vertex);
                  o.uv = v.uv;
                  return o;
            }
            fixed4 frag(v2f i) : SV_Target
            {
                  half4 offset = _MainTex_TexelSize.xxxy *
                        half4(1, 2, 3, 0) * _Offset;
                  half4 scale = half4(1, 1, 1, 1);
                  scale.z *= (1.0 - 1.0 / _Power);
                  scale.y *= scale.z * (1.0 - 1.0 / _Power);
                  scale.x *= scale.y * (1.0 - 1.0 / _Power);
                  fixed4 color = (tex2D(_MainTex, i.uv + offset.zw) *
                        scale.x + tex2D(_MainTex, i.uv + offset.yw)
                        * scale.y + tex2D(_MainTex, i.uv + offset.xw)
                        * scale.z + tex2D(_MainTex, i.uv) +
                        tex2D(_MainTex, i.uv - offset.xw) * scale.z +
                        tex2D(_MainTex, i.uv - offset.yw) * scale.y +
                        tex2D(_MainTex, i.uv - offset.zw) * scale.x)
                        / (scale.x * 2 + scale.y * 2 + scale.z * 2 +
                        scale.w);
                  return fixed4(color.rgb, 1);
            }
            ENDCG
      }
//another pass for y-blur
}
Fallback off
}
```

Вхідними даними шейдера є текстура та два дійсних числа. Як текстура використовується поточна рендер-текстура що створюється камерою. Значення _Power відповідає за силу розмиття, а значення _Offset за інтервал між пікселями, що використовуються для змішування. Так як цей шейдер використовуються як ефект камери то виставлені відповідні значення, такі як запис у буфер глибини відключена, тест на відстань від камери відключений та відключене забраковування матеріалу по його орієнтації до камери. У проході шейдера використовується стандартна вершинна функція, яка лише передає інформацію до фрагментної функції. У фрагментній функції використовується метод експоненційного зменшення ваги сусідніх пікселів. Тому кожен наступний піксель, що знаходиться на інтервалі визначеному у вхідних даних, буде мати вагу меншу за попередній в певну кількість разів. В результаті при

найсильнішому розмитті коефіцієнти матриці будуть такими як наведено у табл. 3.1.

Таблиця 3.1

Матриця змішування

0.0020	0.0041	0.0082	0.0165	0.0082	0.0041	0.0020
0.0041	0.0082	0.0165	0.0330	0.0165	0.0082	0.0041
0.0082	0.0165	0.0330	0.0661	0.0330	0.0165	0.0082
0.0165	0.0330	0.0661	0.1322	0.0661	0.0330	0.0165
0.0082	0.0165	0.0330	0.0661	0.0330	0.0165	0.0082
0.0041	0.0082	0.0165	0.0330	0.0165	0.0082	0.0041
0.0020	0.0041	0.0082	0.0165	0.0082	0.0041	0.0020

Як видно у таблиці центральне значення переважає над усіма, і значення зменшуються у два рази за кожен крок по таблиці у напрямку від центру. Ця таблиця утворилася внаслідок двох проходів шейдера, один з них використовує лише горизонтальні значення для змішування, інший же лише вертикальні, звідси і з’явилася така матриця пірамідального вигляду. Для покращення якості зображення таке змішування відбувається двічі, так як друге змішування підтягує собою пікселі що знаходяться ще далі як за 3 інтервали від центру. А щоб прискорити обробку зображення використовується його зменшена копія, яка за допомогою алгоритму подвійної або потрійної фільтрації змішує сусідні пікселі зменшуючи їх кількість.

2.4. Висновки до розділу 2

Для покращення роботи, зовнішнього вигляду та додаткового функціоналу було розроблено кілька алгоритмів. Алгоритм роботи з гіроскопом імітує доповнену реальність тоді коли її реалізація неможлива з ряду причин, таких як втрата маркеру з поля зору камери. Алгоритм масштабування карти жестами був розроблений для зручного користування інтерактивною картою що допомагає користувачу орієнтуватися по музею. А також алгоритм розмиття заднього фону що покращує зовнішній вигляд програми за допомогою ефектної появи спливаючих вікон.

3. РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДОДАТКУ ДОПОВНЕНОЇ РЕАЛЬНОСТІ

3.1. Архітектура додатку

В даному додатку використовується MVCS або Model View Controller Service архітектура, де окрім звичної MVC моделі використовуються ще сервіси, роль яких другорядна і їх робота доповнює додаток додатковими можливостями.

У ролі Model виступають класи які використовуються для групування дрібних даних, абстрагування та передачі їх між обробниками.

View відповідає за відображення даних як на екрані так і для сторонніх ресурсів. Також View є компонентом ігрового об’єкту, тобто ініціалізований об’єкт цього класу міститься в ієрархії сцени. Це дозволяє швидко працювати з ієрархією сцени та об’єктами які пов’язані логікою з цим компонентом.

Controller це ядро системи що відповідає за обробку моделей та їх даних, керування основними об’єктами на сцені та управління логікою системи. Для того щоб дати коду можливість масштабування контроллер виступає у ролі шаблона Команда, де кожна команда виконує свою роль і нічого більше. Таку архітектуру простіше масштабувати і тестувати так як додавання нового коду не змінює реалізацію старого і визначення помилок зводиться до визначення команди у якій сталася помилка.

Service – об’єкти основна роль яких винести код що дублюється окремо або надати додатку розширені можливості, наприклад сервіс завантаження даних з інтернету чи сервіс роботи з нативним аудіо-плеєром. Також сервіси можна використовувати для створення груп моделей як Фабрики. Такі об’єкти сильно відділені від усієї системи за допомогою інтерфейсів що допомагає легко їх замінити у разі потреби [5].

Ядро системи побудоване на кількох шаблонах проектування. Найбільшу роль серед них відіграє Фасад (Dispatcher). Його функція – зробити якісно доступним функції інших об’єктів та виконання команд з будь-якої точки коду. Це досягається за допомогою прив’язування об’єктів команд до певного значення, за допомогою якого і відбувається виклик команди Фасадом. Також Фасад дозволяє робити виклик будь-яких подій у вигляді значень, на які можуть підписатися будь-які об’єкти. В такому разі вони будуть проінформовані у разі виклику команди або звичайної події. Для того щоб прив’язати подію до команди використовується словник [19].

У лістингу 3.1 наведений код одного з методів фасаду.

Лістинг 3.1. Реалізація методу Dispatch

```
public void Dispatch (object eventType, object data)
{
IEvent evt = conformDataToEvent (eventType, data);
if (evt is IPoolable)
{
      (evt as IPoolable).Retain();
}
bool continueDispatch = true;
if (triggerClients != null)
{
      isTriggeringClients = true;
      foreach (ITriggerable trigger in triggerClients)
      {
            try
            {
                  if (!trigger.Trigger(evt.type, evt))
                  {
                        continueDispatch = false;
                        break;
                  }
            }
            catch (Exception ex)
            {
                  internalReleaseEvent(evt);
                  throw (ex);
            }

      }
      if (triggerClientRemovals != null)
      {
            flushRemovals();
      }
      isTriggeringClients = false;
}
if (!continueDispatch)
{
      internalReleaseEvent (evt);
```

```
        return;
    }
    IEventBinding binding = GetBinding (evt.type) as IEventBinding;
    if (binding == null)
    {
        internalReleaseEvent (evt);
        return;
    }
    object[] callbacks = (binding.value as object[]).Clone() as object[];
    if (callbacks == null)
    {
        internalReleaseEvent (evt);
        return;
    }
    for(int a = 0; a < callbacks.Length; a++)
    {
        object callback = callbacks[a];
        if(callback == null)
            continue;
        callbacks[a] = null;
        object[] currentCallbacks = binding.value as object[];
        if(Array.IndexOf(currentCallbacks, callback) == -1)
            continue;
        if (callback is EventCallback)
        {
            invokeEventCallback (evt, callback as EventCallback);
        }
        else if (callback is EmptyCallback)
        {
            (callback as EmptyCallback)();
        }
    }
    internalReleaseEvent (evt);
    }
```

Роль цього методу – викликати всі функції підписані на відповідну подію та відповідні списки команд. Для початку потрібно створити об’єкт що буде передавати вкладені дані між ними. Потім дістати прив’язаний до події список команд і виконати його. В кінці виконати підписані на цю подію функції. Цей метод дозволяє підтримувати зв’язок між різними компонентами програми без їх взаємних посилань що в свою чергу не заплутує проект.

Наступним важливим компонентом цієї архітектури є Dependency Injector – шаблон що розвинений із відомого шаблона проектування Singleton. Його основна роль – надати доступ до основних компонентів системи за допомогою їх ін’єкції у цей об’єкт. Ін’єкція відбувається під час створення об’єкту. За допомогою стандартної бібліотеки C# System.Reflection усі об’єкти що мають поле у вигляді Property будь-якого

типу з атрибутом Injection обробляються, після чого це поле вказує на відповідний об’єкт відповідного типу що зберігається у словнику ін’єктора. Така структура зменшує кількість статичних полів та у коді виглядає як підключення нового функціоналу. Це зменшує використання популярного шаблону Singleton який сильно заплутує код якщо ним неправильно користуватися [19].

Лістинг 3.2. Реалізація ін’єкції

```
public object Instantiate(IInjectionBinding binding, bool
tryInjectHere)
{
failIf(binder == null, "Attempt to instantiate from Injector without
a Binder", InjectionExceptionType.NO_BINDER);
failIf(factory == null, "Attempt to inject into Injector without a
Factory", InjectionExceptionType.NO_FACTORY);
armorAgainstInfiniteLoops (binding);
object retv = null;
Type reflectionType = null;
if (binding.value is Type)
{
      reflectionType = binding.value as Type;
}
else if (binding.value == null)
{
      object[] tl = binding.key as object[];
      reflectionType = tl [0] as Type;
      if (reflectionType.IsPrimitive || reflectionType ==
typeof(Decimal) || reflectionType == typeof(string))
      {
            retv = binding.value;
      }
}
else
{
      retv = binding.value;
}
if (retv == null)
{
      IReflectedClass reflection = reflector.Get (reflectionType);
      Type[] parameterTypes = reflection.constructorParameters;
      object[] parameterNames = reflection.ConstructorParameterNames;
      int aa = parameterTypes.Length;
      object[] args = new object [aa];
      for (int a = 0; a < aa; a++)
      {
            args [a] = getValueInjection (parameterTypes[a] as Type,
parameterNames[a], reflectionType, null);
      }
      retv = factory.Get (binding, args);
      if (tryInjectHere)
      {
            TryInject(binding, retv);
      }
}
```

```
infinityLock = null;
return retv;
}
```

У лістингу 3.2 описаний метод ін’єкції за допомогою рефлексії. Спочатку відбувається перевірка на присутність всіх компонентів і на можливе зациклення ін’єкцій. Потім з прив’язки дістаємо тип, перевіряємо чи такий об’єкт вже створено, якщо ні то створюємо його. І після цього вставляєм цей об’єкт в те місце де був викликаний атрибут.

Для відділення View від Controller використовується шаблон проектування Посередник або Mediator. Його роль відділити реалізацію View від іншого коду. Зазвичай якщо цю проблему не вирішувати то вона спричиняє ряд непередбачуваних помилок коли на сцені є кілька незалежних View що працюють одна з одною. У такому разі якщо їх спілкування не налаштоване на вищому рівні то при видаленні однієї View інші можуть працювати неправильно. Тому така група розростається і з часом приходиться рефакторити код. Для того щоб уникнути створення заплутаних груп коду і розділити функціонал використовується Посередник, роль якого спілкування з іншими об’єктами.

Лістинг 3.3. Реалізація зв’язку Mediator та View

```
virtual protected void bubbleToContext(MonoBehaviour view, BubbleType
type, bool finalTry)
{
    IContext context = MainContextView.Instance.context;
    bool success = true;
    switch (type)
    {
        case BubbleType.Add:
            context.AddView(view);
            registeredWithContext = true;
            break;
        case BubbleType.Remove:
            context.RemoveView(view);
            break;
        case BubbleType.Enable:
            context.EnableView(view);
            break;
        case BubbleType.Disable:
            context.DisableView(view);
            break;
        default:
            success = false;
```

```
                break;
        }
        if (success)
        {
            return;
        }
    }
```

У лістингу 3.3 описана частина коду яка викликає відповідні події про появу, видалення, активацію та деактивацію View до медіатора.

Ядро системи і її поведінка описується за допомогою шаблону Стратегія або Context. При запуску системи виконується код стратегії, як буде працювати система. Основна роль шаблону – зв’язати команди з подіями, згрупувати команди в речення, заповнити Dependency Injector об’єктами, зв’язати класи View з класами посередників. Такий підхід дозволяє швидко видалити функціонал чи замінити його іншим, обирати функціонал в залежності від певних змінних середовища чи завантаженого конфігураційного файлу, наприклад прив’язувати різні команди до однієї події в залежності від платформи на якій виконується код.

3.2. Інтерфейс користувача

Інтерфейс користувача в додатку з доповненою реальністю можна розділити на дві частини, де перша – основна, де працює доповнена реальність і друга – додаткова, для налаштувань і додаткового контенту.

Перша частина обов’язково має містити зображення з камери, або бути з ним зв’язаною. Для того щоб не перекривати зображення з камери і дозволити користувачу повністю відчути зв’язок віртуальності і реальності у цьому проекті було вирішено використати мінімальний користувацький інтерфейс. Основні кнопки управління анімаційним контентом і доповненою реальністю вирішено винести на панель інструментів, яка автоматично покидає екранний простір. Основні інформаційні повідомлення такі як інформація по використанню для зручності появлятимуться як окреме вікно з кнопкою “закрити”.

Друга частина користувацького інтерфейсу відповідає за управління додатком і його додатковим функціоналом що не зв’язаний з доповненою реальністю. Це такі вікна як вікно налаштувань, вибору мови, завантаження анімаційного контенту, головне меню та зворотнього зв’язку з розробниками. Ця частина користувацького інтерфейсу має бути максимально простою і доступ до основного функціоналу має відбуватися швидко і просто.

Порядок у якому користувач буде використовувати додаток такий:

1. Меню завантаження (завантаження конфігураційного файлу і налаштування додатку).
2. Якщо користувач відкриває додаток вперше то відкриється меню вибору мови і потім меню завантаження анімаційного контенту, де користувач може обрати той набір анімацій що відповідає музею в якому він знаходиться.
3. Головне меню із можливими переходами в меню камери, та меню налаштувань.
4. Якщо користувач перейде у меню камери то побачить спочатку інструкції по використанню доповненої реальності та додаткового функціоналу.
5. У меню камери користувач може відсканувати маркер, після чого завантажеться відповідний анімаційний контент та прикріпиться на маркер. Якщо сканер втратить маркер з поля зору то позиція контенту буде зміщена в центр камери на відповідну відстань яку можна буде регулювати, а його орієнтація буде обраховуватися відносно реальної орієнтації девайсу так, щоб їх вертикальні осі співпадали.
6. Також у меню камери користувач може користуватися додатковим функціоналом, таким як мапа музею та можливістю зробити фотознімок.

3.3. Модуль завантаження віддаленого контенту

Модуль завантаження віддаленого контенту складається з двох підмодулів – сам сервіс завантаження та віддалене хранилище даних.

Сервіс завантаження віддаленого контенту – це окремий модуль програми який максимально не зв’язаний з реалізацією інших модулів і використовується через інтерфейс у стратегії виконання програми і у деяких командах що напряму залежать від функції завантаження. Сервіс складається з класу створення і обробки запитів, команд що запускають відповідні типи запитів та відповідні їм моделі даних для передачі інформації в команди і в сервіс завантаження.

Мета розробки цього сервісу – масштабування додатку, а саме:

1. Так, як частина ресурсів ПЗ знаходяться на віддаленому сервері то контент можна змінювати не випускаючи нових версій ПЗ.
2. Спрощується тестування додатку, так як новий анімаційний контент можна відразу ж завантажити на сервер і протестувати на мобільному додатку, а не лише в середовищі розробки.
3. Зменшення розміру додатку, так як користувачу не потрібно завантажувати всі ресурси для коректної роботи додатку.
4. Швидке виправлення несправних ресурсів, з мінімальним залученням до роботи розробників.

Наведемо приклад кількох класів та методів цього сервісу та пояснимо їх роботу.

Лістинг 3.4. Команда завантаження файлу

```
public class NetworkLoadFileCommand : BaseCommand {
    [Inject]
    public INetworkService networkService { get; set; }
    private FileRequestModel request;
    private int errorCounter = 5;
    private float startRequestTime;

    public override void Execute()
    {
        request = eventData.data as FileRequestModel;
        if (request == null)
        {
```

```
            Debug.LogError("NetworkLoadFileCommand: eventData.data !=
FileRequestModel");
            Fail();
            return;
        }
        Retain();
        startRequestTime = Time.time;
        SendRequest();
    }

    private void SendRequest()
    {
        networkService.SendRequest(request.URL, OnResponse,
ShowProgress, HTTPMethod.Get, request.Data, request.Headers,
priority: request.priority);
    }

    private void OnResponse(WWW www)
    {
        request.www = www;
        if (!string.IsNullOrEmpty(www.error))
        {
            if (--errorCounter <= 0)
            {
                if (!string.IsNullOrEmpty(www.text))
                {
                    var json = SimpleJSON.JSON.Parse(www.text);
                    if (json["status"].AsInt == 500)
                    {
                        Debug.LogWarning(www.text);
                        request.Callback();
                        Release();
                        return;
                    }
                }
                request.Error(www);
                Release();
                Fail();
                return;
            }
            else
            {
                SendRequest();
            }
        }
        else
        {
            request.SetFile(www.bytes);
            request.Callback();
            www.Dispose();
            Release();
        }
    }
}
```

У лістингу 3.4 наведена команда завантаження файлу. Порядок її виконання такий – спочатку відбувається ін’єкція сервісу в команду, потім перевірка вхідних даних і виклик методу SendRequest з відповідними

параметрами, а саме метод Http запиту – GET, та відповідні значення що встановлені в моделі запиту, наприклад пріоритет, хедери, дані, колбек-функції для зворотнього виклику для обробки відповіді, помилки і прогресу завантаження. У команді є лічильник помилок, його значення складає 5 на початку і у разі помилки зменшується. Коли значення досягне 0 тоді відбудеться виклик колбек-функції помилки і команда завершить свою роботу, інакше завантаження розпочнеться спочатку. Це зроблено з метою спростити завантаження через мобільну мережу, що може викликати перебої в завантаженні. При успішному завантаженні видобуваються дані із відповіді та передаються в модель, після чого викликається колбек-функція успішного завантаження.

Лістинг 3.5. Модель завантаження файлу

```
public class FileRequestModel : IDisposable {
    public byte[] rawFile = null;
    private System.Action<FileRequestModel> callback;
    private System.Action<float> progressCallback;
    private System.Action<string> errorHandler;
    public ThreadPriority priority;
    public JSONNode Data { get; private set; }
    public Dictionary<string, string> Headers { get; private set; }
    public string URL { get; private set; }
    public WWW www = null;
    public float Progress { get; private set; }

    public FileRequestModel(string url,
System.Action<FileRequestModel> callback, System.Action<float>
progressCallback = null, JSONNode data = null, Dictionary<string,
string> headers = null, System.Action<string> errorHandler = null,
ThreadPriority priority = ThreadPriority.Normal)
    {
        this.callback = callback;
        this.errorHandler = errorHandler;
        this.progressCallback = progressCallback;
        this.priority = priority;
        URL = url;
        Data = data;
        Headers = headers;
    }

    public void SetFile(byte[] file)
    {
        rawFile = file;
    }

    public void ProgressCallback(float progress)
    {
        Progress = progress;
```

```
            if (progressCallback != null)
            {
                progressCallback(progress);
            }
        }

        public void Callback()
        {
            callback(this);
        }

        public void Error(WWW www)
        {
            if (errorHandler != null)
            {
                errorHandler(www.error);
            }
        }
    }
```

У лістингу 3.5 наведено код моделі завантаження файлу. Основна її функція – передати необхідні для завантаження дані з команди де відбувається виклик завантаження до команди що це завантаження обробляє, а також забезпечити зворотній зв’язок через колбек-функції. У випадку колбек-функції успішного завантаження як параметр використовується сама модель, що є зручним для обробки всієї інформації що брала участь у запиті.

Лістинг 3.6. Сервіс завантаження ресурсів

```
public class NetworkService : INetworkService
{
    [Inject] public IExecutor Executor { get; set; }
    public const string API_KEY = "****";
    public List<WWW> requests = new List<WWW>();
    public Dictionary<int, Coroutine> coroutines = new
Dictionary<int, Coroutine>();
    private Coroutine request;
    int counter = 0;
    public void SendRequest(
        string url, Action<WWW> callback,
        Action<float> progressCallback = null,
        HTTPMethod method = HTTPMethod.Get,
        JSONNode data = null,
        Dictionary<string, string> headers = null,
        ThreadPriority priority = ThreadPriority.Normal,
        bool useAPIKey = true)
    {
        int id = counter++;
        request = Executor.Execute(SendRequestCoroutine(id, url,
callback, progressCallback, method, data, headers, priority,
useAPIKey));
        coroutines.Add(id, request);
```

```
    }
    public IEnumerator SendRequestCoroutine(
        int id, string url, Action<WWW> callback,
        Action<float> progressCallback = null,
        HTTPMethod method = HTTPMethod.Get,
        JSONNode data = null,
        Dictionary<string, string> headers = null,
        ThreadPriority priority = ThreadPriority.Normal,
        bool useAPIKey = true)
    {
        WWW www = null;
        if (data == null)
        {
            data = new JSONClass();
        }
        string finalEscapeUriString = "";
        finalEscapeUriString = System.Uri.EscapeUriString(url);
        finalEscapeUriString += BuildUrlEncodedPostData(null,
useAPIKey);
        if (headers == null)
        {
            headers = new Dictionary<string, string>();
        }
        www = new WWW(finalEscapeUriString);
        www.threadPriority = priority;
        requests.Add(www);
        while (www != null && string.IsNullOrEmpty(www.error) &&
!www.isDone)
        {
            if (progressCallback != null)
            {
                progressCallback(www.progress);
            }
            yield return null;
        }
        if (www != null)
        {
            callback(www);
            requests.Remove(www);
        }
        coroutines.Remove(id);
    }
}
```

Скрипт наведений у лістигну 3.6 відповідає власне за паралельне відправлення та обробку кількох запитів. Відкритих методів два, один з них запускає співпрограму, яка працює над одним запитом і чекає його завершення паралельно до основної роботи додатку. Інший метод слугує для того щоб зупинити виконання співпрограм і завершити надсилання запитів і викликається у разі помилки завантаження. У сервісі міститься словник співпрограм та їх індексів та список запитів що зараз виконуються для того щоб мати змогу всіх їх завершити в один момент. Спочатку

створюється співпрограма, їй призначається унікальний номер який передається як атрибут у співпрограму та посилання на співпрограму зберігається у словнику. У співпрограмі створюється запит і посилання на нього зберігається у списку. В кінці виконання співпрограми посилання на запит видаляється зі списку а за допомогою унікального ідентифікатора співпрограми з словника видаляється посилання на співпрограму, після чого співпрограма завершує свою роботу.

3.4. Особливості реалізації додатку доповненої реальності

Додаток доповненої реальності повинен містити ряд особливостей які потрібно реалізувати для коректної його роботи та зрозумілого переходу між станами програми.

Першою з таких особливостей є вирішення проблеми втрати маркера з поля зору камери. У цьому додатку реалізований функціонал перенесення анімаційного контенту у позицію перед камерою і використання гіроскопу для його орієнтування вертикально відносно реальної вертикалі.

Другою особливістю є питання завантаження даних про маркери. Якби проект не передбачав якість масштабування то зберігання даних про маркери в статичних ресурсах і завантаження їх за замовчуванням на початку роботи програми було б простим рішенням. Але цей додаток передбачає можливість масштабування тому необхідно створити модуль динамічного завантаження маркерів.

3.4.1. *Оцінка якості розпізнавання маркерів*

Наведемо ряд особливостей маркерів [2]:

1. Якість розпізнавання не залежить від того чи використовується чорно-білий варіант зображення чи варіант у кольорі.
2. Якість розпізнавання не залежить від розміру зображення, якщо розмір перевищує відмітку у 400*400 пікселів.

3. Для того щоб зробити частину зображення не розпізнаваною варто її розмити одним із засобів будь-якого редактора зображень.
4. Максимальний розмір маркеру – 2.5Мб, тому використання JPG формату має переваги над PNG форматом для зменшення розміру ресурсів.
5. Якість маркеру залежить від кількості контурних вузлів знайдених на зображенні.

За допомогою цих особливостей та особистого досвіду сформовано наступні вимоги до маркерів:

1) формат зображення – JPG;
2) розмір зображення 512*512, де одна зі сторін може варіюватися якщо відношення сторін не 1:1;
3) використання чорно-білого формату для зменшення розміру;
4) оцінка зображення на порталі Vuforia не менше 4.

В критеріях вище було вказано оцінку маркера яка має бути для швидкого його розпізнавання. Всього є 5 рівнів оцінки, кожна з яких відображає кількість контурних вузлів на зображенні та їх якість [1].

1. Рівень 1 (рис. 3.1). Відповідає оцінці 0 або 1 на порталі Vuforia, у такому випадку зображення неякісне, розмите і має дуже мало контурних вузлів, що робить його майже неможливим для розпізнавання.
2. Наступний рівень відповідає оцінці 2, та містить більше контурних вузлів, погано розпізнається сканером і не рекомендується для використання.
3. Рівень 3 (рис. 3.2). Це мінімальний рівень для середніх показників стабільності роботи, може використовуватися для розпізнавання.
4. Рівень 4 (рис. 3.4). Досягають більшість використовуваних маркерів у цій сфері. При попаданні хоча б 40% маркера у поле зору сканера він розпізнається.

5. Рівень 5 (рис. 3.3). Визначається дуже високими показниками розпізнаваності. Навіть при поганому освітленні або попаданні 20% маркера у поле зору сканера він розпізнається.

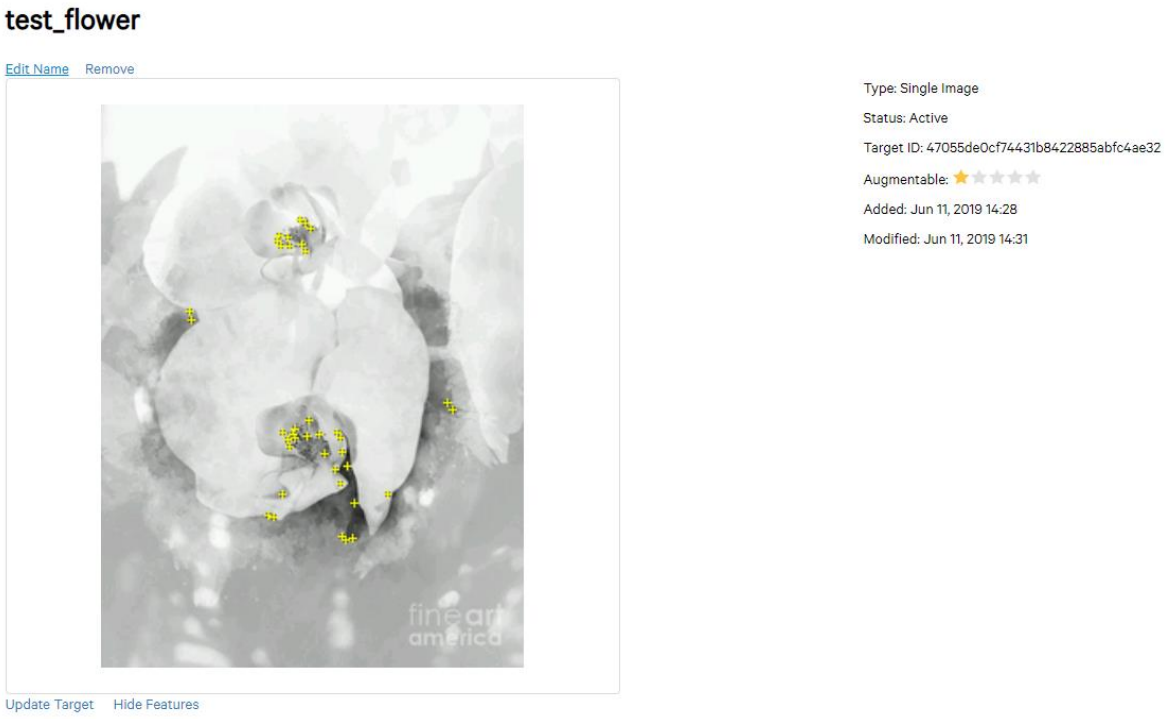


Рис. 3.1. Оригінальний маркер

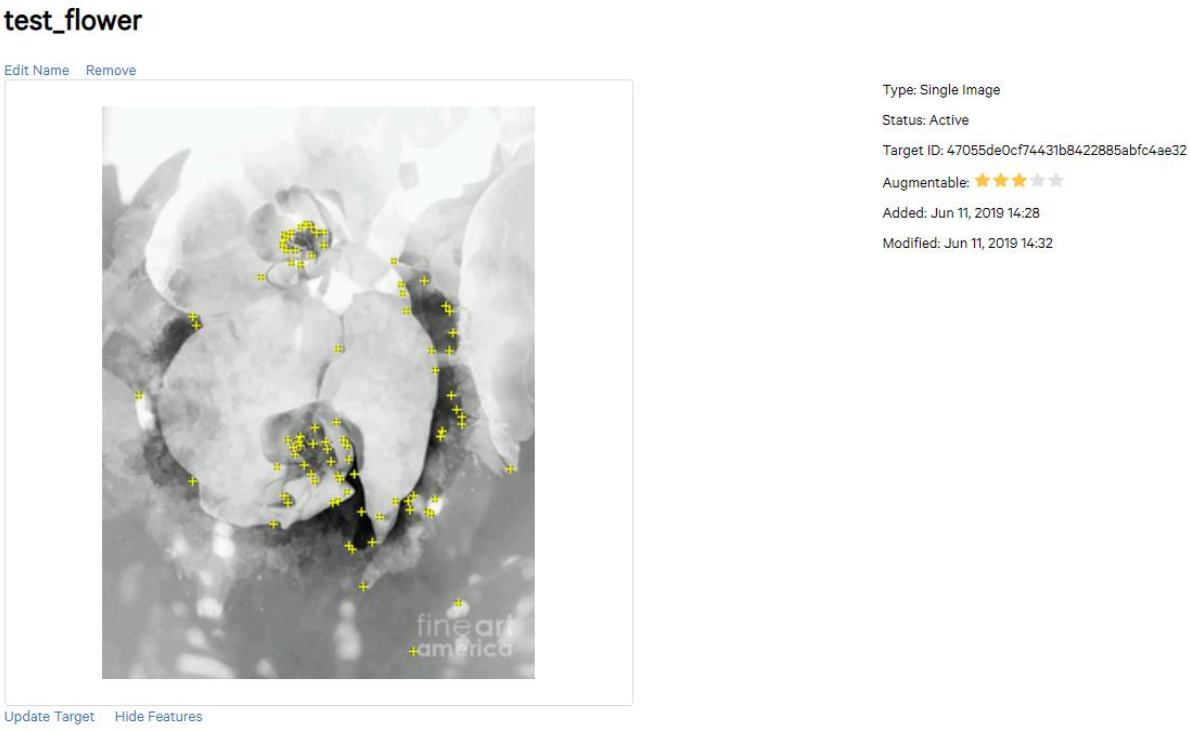


Рис. 3.2. Покращений маркер шляхом підвищення контрасту

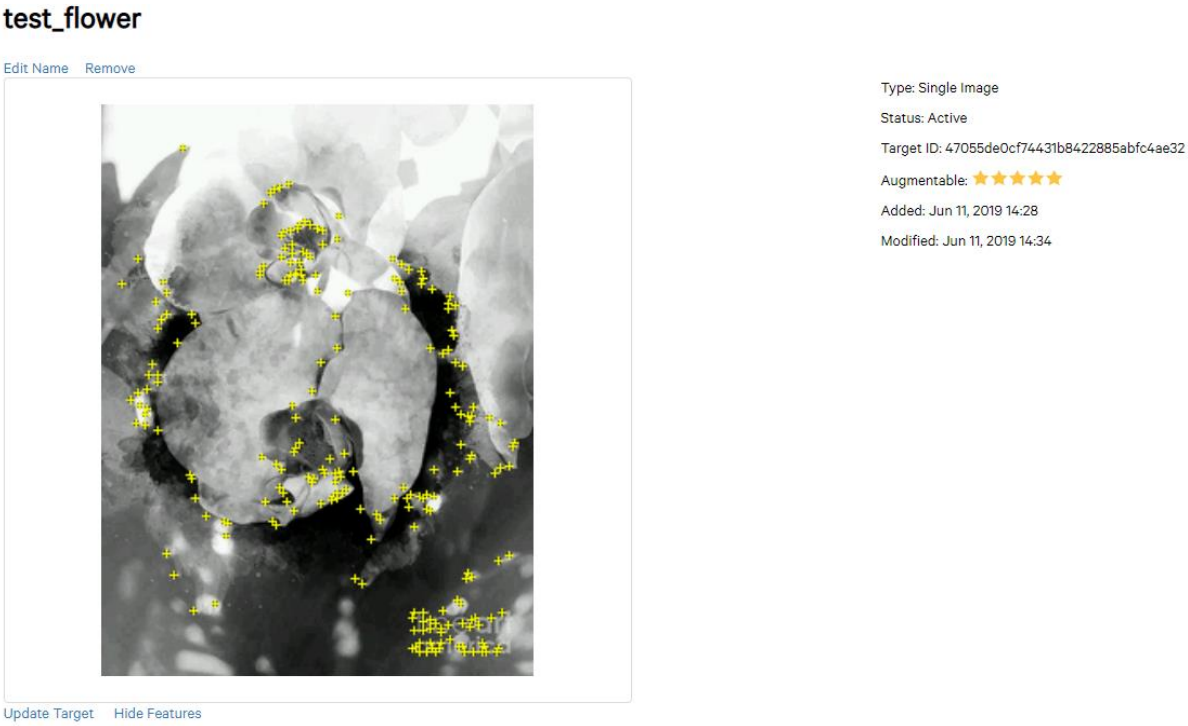


Рис. 3.3. Покращений маркер шляхом гамма-корекції

Рис. 3.4. Покращений маркер шляхом додавання геометричних об’єктів

Щодо покращення оцінки маркера то можна використовувати кілька методів. На рис. 3.1 зображено оригінальне зображення, на рис. 3.2 зображення після збільшення контрастності та гамма-корекції, після чого перехід між кольорами став різкішим, саме це збільшило кількість вузлових точок та загальну оцінку маркера. На рис. 3.3 зображено маркер з найкращою оцінкою 5. Для досягнення цього був використаний попередній метод ще раз, але з більшими параметрами контрастності та гамма-корекції. Як можна побачити це сильно змінює оригінальне зображення, тому запропоновано ще один спосіб покращення оцінки, а саме додавання картинок різних геометричних об’єктів, до зображення. Як наслідок оцінку було покращено з 1 до 4. Було використано геометричні об’єкти різного кольору, тому що їх перетин формує чітку вузлову точку. Але для цього не варто використовувати однакові геометричну об’єкти тому що можуть відбуватися перебої в скануванні. Основною причиною цього буде сканування невідповідної частини зображення.

Варто додати те, що на різних маркерах не має бути спільних об’єктів, тому що виникають конфлікти при скануванні, внаслідок чого один маркер може сприйматися за інший.

3.5. Висновки до розділу 3

Розроблення програмного продукту базувалося на архітектурі MVCS що забезпечує розділеність реалізацій модулів програми між собою. Архітектурний підхід сильно спростив задачу написання основного та додаткового функціоналу так як кожен функціонал існував як окремий модуль і не посилався на функціонал інших модулів і не використовував їх напряму. За допомогою таких структурних та шаблонів поведінки як Dispatcher, Command, Mediator, Context, Dependency Injection було реалізовано спілкування між класами і модулями програми. Модуль віддаленого завантаження анімаційного контенту вирішує проблему розростання розміру додатку та його майбутнє масштабування. Тому для

додавання додаткового контенту не потрібно створювати нову версію додатку. Також було розглянуто критерії оцінювання, способи покращення маркерів та сформовано вимоги до зображень які використовуватимуться у якості маркерів.

4. АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

4.1. Критерії визначення якості продукту та аналіз розробленого продукту

В ході аналізу, розробки і тестування продукту було визначено такі критерії якості:

1) час завантаження режиму камери менше 5 секунд;
2) стійкість до перебійного режиму роботи мережі інтернет;
3) миттєве сканування маркера коли він повністю в полі зору сканера і займає не менше 35% простору камери;
4) сканування маркера якщо більше 60% його ефективної площі потрапляє у поле зору сканера;
5) сканування маркера якщо він займає хоча б 20% простору камери.

Протестувавши продукт можна зауважити кілька відхилень від критеріїв. Перше відхилення це той факт що починаючи з версії Vuforia 8.1 під час першого завантаження режиму камери за наявності інтернету завантажується інформація про девайс та поточну камеру із ресурсів Vuforia для кращої роботи з камерою. Ця функція корисна для продукту тому було прийнято рішення не вимикати її.

Друге відхилення це сам факт якості маркерів який впливає на якість їх розпізнавання. Тому якщо маркер поганої якості то і критерії щодо нього мають бути спрощеними. Сам факт сканування маркера з оцінкою ⅗ буде достатнім критерієм.

Тестування проводилося димовим методом. Мета методу – знайти основні проблеми, які спричиняють неправильну роботу додатку, проаналізувати та виправити їх. В цьому методі використовувалося ручне тестування, за якого визначалися основні проблеми роботи технології доповненої реальності.

В результаті тестування були визначені основні проблеми, серед яких найважливіша це неправильна орієнтація об’єкту під час його відслідковування. Вирішення цієї проблеми зайняло б дуже багато часу, так як знайти підхід до правильної орієнтації за такої проблеми одночасно не змінюючи правильність виконання за звичайних умов дуже важко. Тому було вирішено не вирішувати цю проблему і дочекатися стабільнішої версії технології доповненої реальності.

Для тестування продуктивності було використано вбудований в рушій інструмент Unity Profiler та Android Debug Bridge. Unity Profiler дозволяє переглянути статистику використання ресурсів таких як CPU Time, GPU Time, Memory. Android Debug Bridge дозволяє переглянути логи виконання програми на мобільному девайсі та зв’язатися з Unity для передачі інформації про продуктивність у Unity Profiler.

В результаті тестування було виявлено що додаток працює належним чином, продуктивність зазвичай на рівні 30 FPS, що є хорошим показником та рівень використання ресурсів не переходить недопустимий рівень.

4.2. Порівняльний аналіз продукту доповненої реальності

Для порівняння потрібно обрати додаток доповненої реальності схожої категорії та цільового ринку. У даному випадку цей продукт спрямований на розвиток і навчання та найбільші емоції може викликати у дітей. Оберемо кілька додатків що схожі на цей за категорією користувачів та типом. Один із таких додатків Hippo Magic. У ньому використовується технологія доповненої реальності що базується на скануванні та трекінгу маркерів. Додаток це доповнення основного продукту яким є книги для дітей віком від 3 до 6 років, які є пізнавальними, наприклад буквар. Суть додатку полягає в тому щоб оживити сторінку книги та відтворити анімаційний контент на сторінці. Для дітей це виглядає як магія, що дуже сильно їх приваблює, про що кажуть висока оцінка на сервісах Google Play та AppStore, хороші відгуки а також висока кількість скачувань. Додаток

використовує схожі ідеї та технології, тому для доцільно було обрати його для порівняння. Отож наведемо кілька критеріїв та порівняємо продукти.

1. Обидва додатки використовують технологію доповненої реальності Vuforia, систему сканування та трекінгу маркерів. Швидкодія однакова.
2. Додатки використовують завантаження анімаційного контенту за допомогою інтернету. Кожна колекція завантажується окремо, але у додатку Hippo Magic це реалізовано зручніше для користувача, тому що немає додаткових меню для вибору колекції для завантаження, натомість завантаження пропонується при першій спробі відсканувати обкладинку книги. Розроблений продукт був спроектований з іншою метою. Не зручно сканувати спеціальний маркер для того щоб завантажити колекцію, так як після завершення роботи програми інформація про останній відвіданий музей вже не є актуальною і подібна реалізація може заплутати користувача. Висновок такий що в обох випадках зручність завантаження досягається різними методами через різне місце розташування маркерів.
3. Користувацький інтерфейс обох продуктів зручний у використанні, але сильно переважає у продукту Hippo Magic через більшу якість дизайну.
4. Додаткові функціональні можливості. Спільні риси це можливість робити фото з камери. В продукті Hippo Magic також є можливість поділитися фото у соцмережах, та зняти відео.
5. Поведінка анімаційного контенту коли маркер втрачено з поля зору. У розробленого продукту наявний алгоритм орієнтації об’єкту вертикально відносно реальної вертикалі, що дає перевагу порівняно з продуктом Hippo Magic, так як частково імітує поведінку об’єкта в доповненій реальності.

Висновок цього порівняння такий, обидва додатки використовують схожі ідеї та технології, але розроблений продукт на фоні продукту Hippo Magic виглядає як прототип що потребує покращення. Основними напрямками поліпшення якості продукту є наповнення його анімаційним контентом та покращення дизайну.

4.3. Перспективи розвитку продукту

Так як продукт на ринку новий та унікальний то за умови його хорошої реальної оцінки та використання в реальних проектах є декілька можливих перспектив розвитку функціональних можливостей що вплине на цінність продукту.

На даному етапі продукт уже може бути використаним кількома музеями без необхідності його розширяти додатковим функціоналом, але є одна умова – анімаційний контент не може містити інтеракції з користувачем чи використання скриптів не визначених у білді. Тому одним з перших розширень може стати система подій та інтеракцій з анімаційним контентом у вигляді поведінкових скриптів, кожен з яких відповідає за обробку однієї події або ініціацію роботи іншого скрипта. Це розширення буде схожим на візуальне програмування, коли маючи набір інструментів та компонентів можна буде налаштувати просту поведінку. Після надання можливості створювати просту поведінку об’єктів у анімаційному контенті не потрібно буде кожного разу при зміненні чи додаванні нової поведінки оновляти додаток на Google Play та AppStore.

Наступна важлива функціональна можливість це додавання модулю аналітики, який буде рахувати кількість сканувань кожного маркера, повного перегляду кожного анімаційного контенту для того щоб визначити який контент користувачам цікавий і який ні, щоб мати можливість замінити його та підняти оцінку продукту.

Ще один важливий пункт для розвитку додатку це його локалізація. Це дозволить потенційним користувачам інших країн його

використовувати, а також розповсюдити його на музеї інших країн. Основна мова додатку англійська – тому це не є проблемою, але локалізація анімаційного контенту зробить додаток доступнішим для більшої аудиторії потенційних користувачів.

4.4. Висновки до розділу 4

У цьому розділі ми оцінили додаток по різним критеріям та порівняли його з уже існуючими прикладами використання доповненої реальності у галузі дитячої освіти, а також розглянули потенційно можливі шляхи розвитку додатку та його популяризацію. Загалом додаток був визначений як зручний у користуванні для людей що стикалися з доповненою реальністю, але для людей що зіткнулись з таким уперше було необхідне додаткове пояснення на питання користування. Тому було запропоновано додати кілька додаткових інструкцій по користуванню. Щодо розвитку додатку то є кілька основних кроків які необхідно зробили щоб покращити роботу і зручність користування додатком, а саме імплементація аналітики, локалізації та додавання нових поведінкових скриптів які дозволять робити інтерактивний анімаційний контент.

ВИСНОВКИ

Головною метою дипломного проекту було створення програмного забезпечення для розширення експозиції музею на основі технології доповненої реальності.

У першому розділі проаналізовано переваги та недоліки графічних рушіїв та технологій доповненої реальності. На основі цього аналізу було обрано кросплатформний графічий рушій Unity Engine та технологію доповненої реальності Vuforia Engine. Для реалізації було використано мову програмування C#.

Для виділення програмного забезпечення серед конкурентів було розроблено ряд алгоритмів наведених у другому розділі. Основну увагу варто звернути на алгоритм орієнтації об’єкту за допомогою гіроскопу, основна мета якого імітувати доповнену реальність, коли технологію доповненої реальності неможливо застосувати через брак ресурсів апаратного забезпечення.

У третьому розділі описано архітектуру програмного забезпечення, її основні модулі та зв’язок між ними. За основний архітектурний шаблон було обрано MVCS Реалізовано модуль завантаження ресурсів з віддаленого сервера, дозволяє масштабувати програмне забезпечення та зручний інтерфейс користувача. Також проведено аналіз зображень що використовуватимуться як маркери для технології доповненої реальності, виділено основні критерії їх оцінки та запропоновані методи покращення їх оцінки. Як додаткові функціональні можливості було запропоновано створити інтерактивну мапу музею та можливість зробити фото та надіслати його у соціальні мережі.

У четвертому розділі проведено аналіз розробленого програмного забезпечення та його тестування. В результаті тестування було знайдено та

вирішено ряд незначних проблем. Як приклад для порівняння було використано аналог розробленого програмного забезпечення з іншою цільовою аудиторією для порівняння продуктивності, зручності користувацького інтерфейсу та реалізації технології доповненої реальності. В результаті порівняння було визначено ряд покращень, серед яких дизайн користувацького інтерфейсу та наповнення віртуальними експозиціями. Також проведено аналіз перспектив розвитку продукту доповненої реальності.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

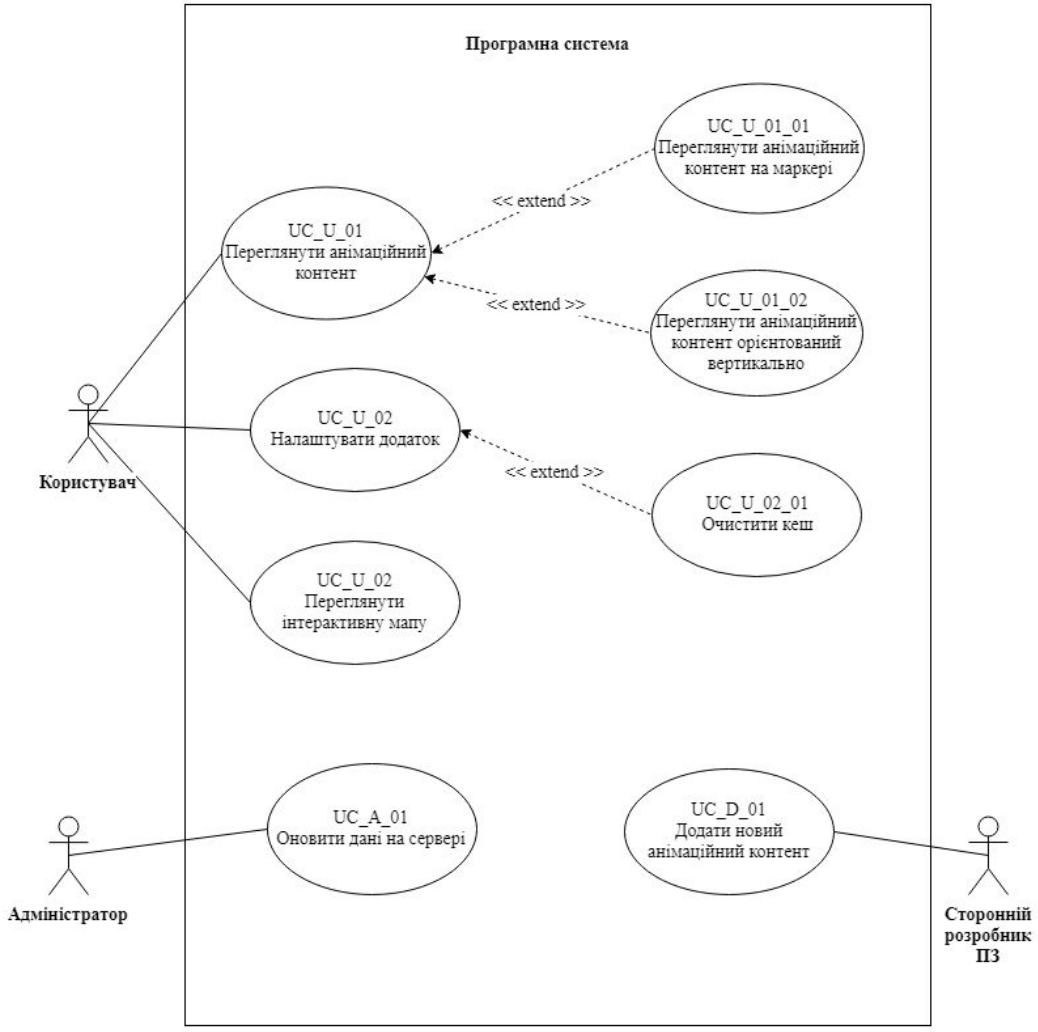
1. Вимоги до розробки спеціалізованих міток для функціонування маркерних додатків з AR-технологією на базі платформи Vuforia [Електронний ресурс]. – Режим доступу: https://moluch.ru/archive/251/57683/. Дата доступу: травень 2019. Назва з екрану.
2. Портал розробників Vuforia Engine [Електронний ресурс]. – Режим доступу: https://developer.vuforia.com. Дата доступу: січень 2019. Назва з екрану.
3. Документація Unity Engine [Електронний ресурс]. – Режим доступу: https://docs.unity3d.com/ScriptReference/index.html. Дата доступу: січень 2019. Назва з екрану.
4. Як доповнена реальність привертає [Електронний ресурс]. – Режим доступу: https://habr.com/ru/company/payonline/blog/399301/. Дата доступу: січень 2019. Назва з екрану.
5. Архітектура MVCS [Електронний ресурс]. – Режим доступу: https://glossar.hs-augsburg.de/Model-View-Controller-Service-Paradigma. Дата доступу: січень 2019. Назва з екрану.
6. Розмивання Гауса [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Gaussian_blur. Дата доступу: травень 2019. Назва з екрану.
7. Blur Postprocessing Effect [Електронний ресурс]. – Режим доступу: https://www.ronja-tutorials.com/2018/08/27/postprocessing-blur.html. Дата доступу: травень 2019. Назва з екрану.
8. Плагін для обміну данних [Електронний ресурс]. – Режим доступу: https://github.com/yasirkula/UnityNativeShare. Дата доступу: травень 2019. Назва з екрану.

9. Кватерніон [Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/183908/. Дата доступу: травень 2019. Назва з екрану.
10. Гіроскоп [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Gyroscope. Дата доступу: травень 2019. Назва з екрану.
11. Загальна інформація про рушій Unity [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Unity_(game_engine). Дата доступу: травень 2019. Назва з екрану.
12. Загальна інформація про рушій Unreal Engine [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/Unreal_Engine. Дата доступу: травень 2019. Назва з екрану.
13. Vuforia [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/Vuforia. Дата доступу: травень 2019. Назва з екрану.
14. Документація ARKit [Електронний ресурс]. – Режим доступу: https://developer.apple.com/documentation/. Дата доступу: травень 2019. Назва з екрану.
15. Android Manifest [Електронний ресурс]. – Режим доступу: https://developer.android.com/guide/topics/manifest/manifest-intro?hl=ru. Дата доступу: травень 2019. Назва з екрану.
16. Що таке ARCore [Електронний ресурс]. – Режим доступу: https://devcolibri.com/what-is-arcore/. Дата доступу: травень 2019. Назва з екрану.
17. Мова програмування C# [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/C_Sharp. Дата доступу: травень 2019. Назва з екрану.

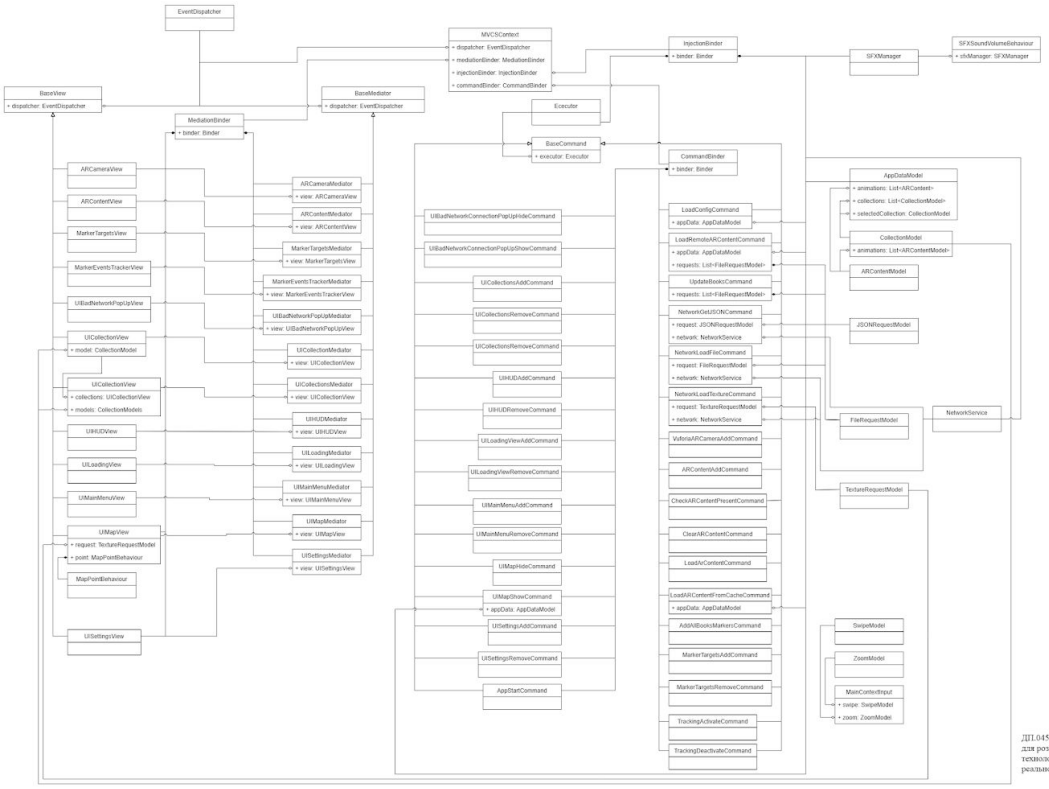
18. Документація C# [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/. Дата доступу: травень 2019. Назва з екрану.
19. Фреймворк StrangeIoC [Електронний ресурс]. – Режим доступу: http://strangeioc.github.io/strangeioc/TheBigStrangeHowTo.html. Дата доступу: травень 2019. Назва з екрану.
20. Документація CG [Електронний ресурс]. – Режим доступу: https://developer.download.nvidia.com/cg/index_stdlib.html. Дата доступу: травень 2019. Назва з екрану.

ДОДАТКИ

Додаток 1
Копії графічних матеріалів

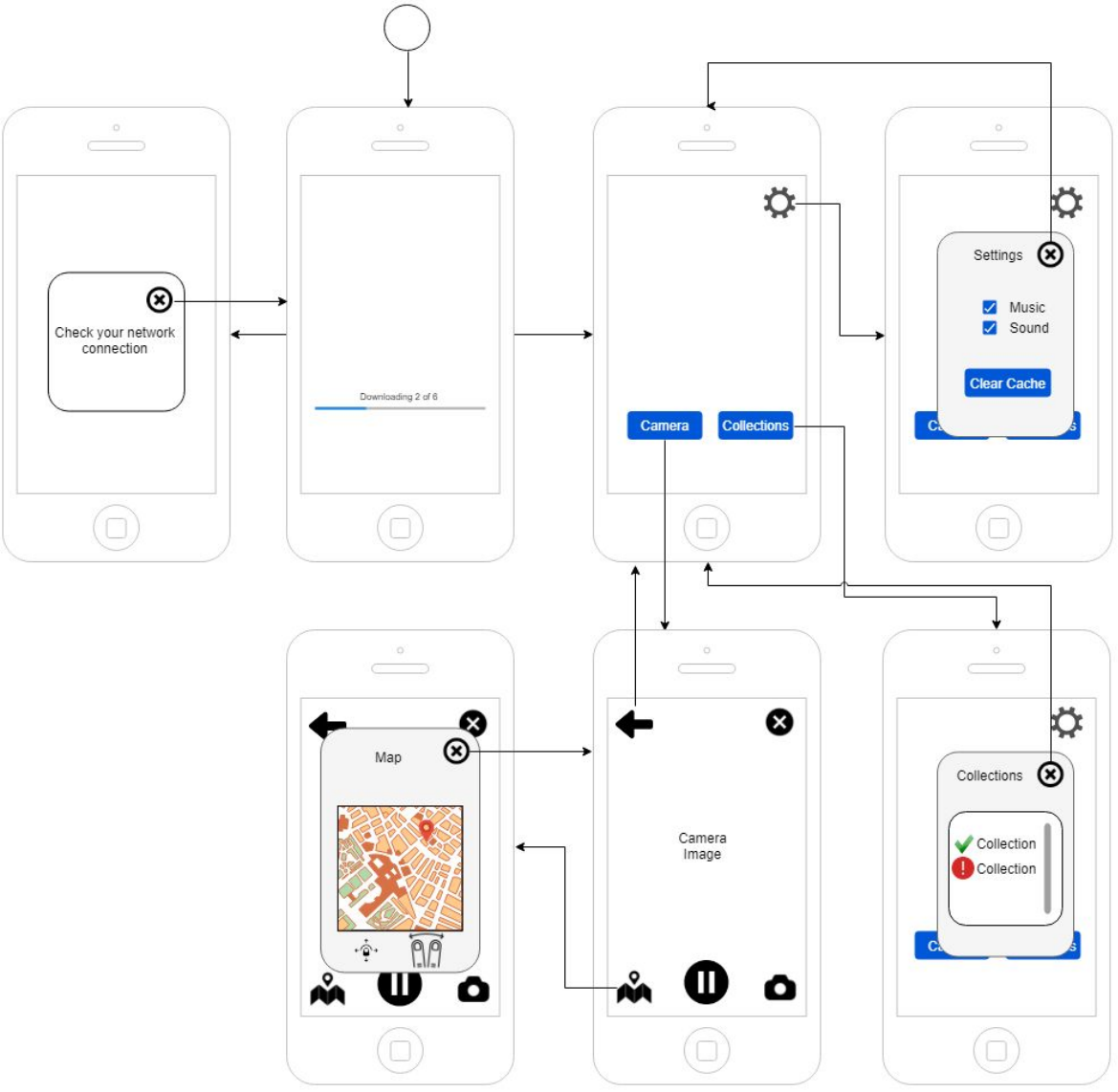


ДП.045480-06-99 Програмне забезпечення для розширення експозиції музею на основі технології доповненої реальності. Функціональні можливості програмного забезпечення. UML діаграма прецедентів

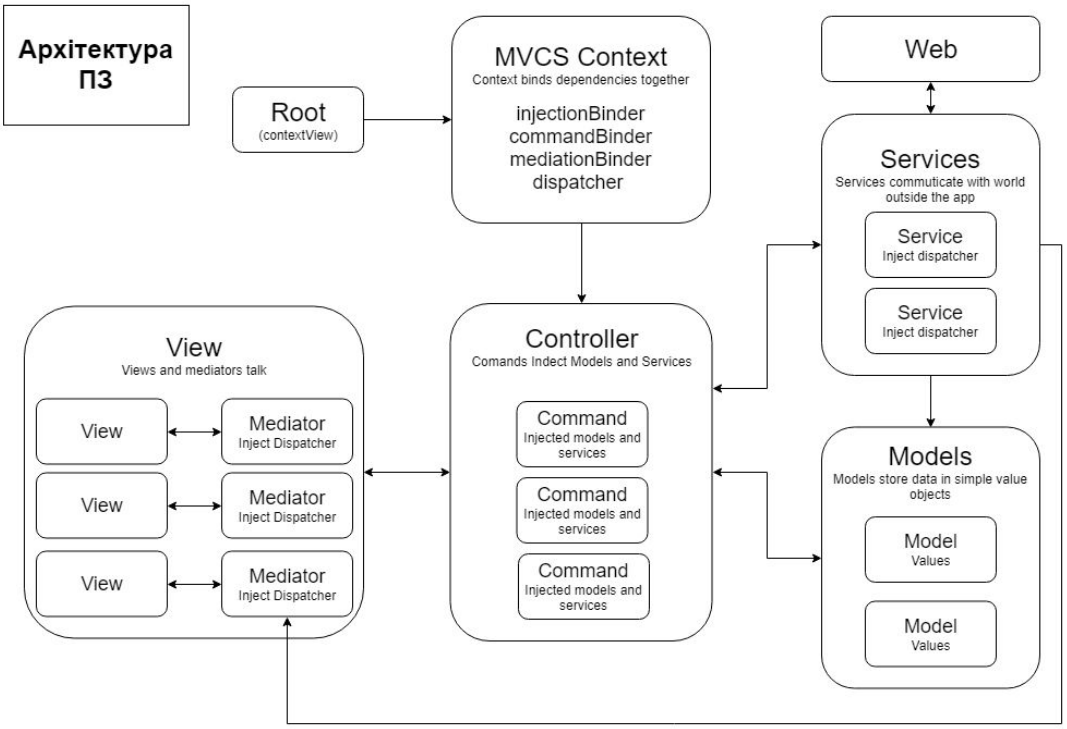


ДП.045480-07-99 Програмне забезпечення для розширення експозиції музею на основі технології доповненої реальності. Архітектура ПЗ. Діаграма класів

UI схема



Саленко А.В. група КП-52



Саленко А.В. група КП-52

Додаток 2
Лістинги програм

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.IO;
using SimpleJSON;

public class LoadConfigCommand : BaseCommand
{

    public string configPath = Path.Combine(Application.persistentDataPath,
"config.json");

    private bool localConfigLoaded = false;
    private bool remoteConfigLoaded = false;

    private Dictionary<int, CollectionModel> booksParseBuffer = new
Dictionary<int, CollectionModel>();
    private Dictionary<int, ARContentModel> animationsParseBuffer = new
Dictionary<int, ARContentModel>();

    private IEnumerable<CollectionModel> booksToUpdate = null;
    private IEnumerable<ARContentModel> animationsToUpdate = null;

    [Inject] public AppDataModel appDataModel { get; set; }

    public override void Execute()
    {
        if (dispatcher.HasListener(Events.E_RetryDownloading, Execute))
        {
            dispatcher.RemoveListener(Events.E_RetryDownloading, Execute);
        }
        LoadLocalConfigs();

        if (localConfigLoaded)
        {
            dispatcher.Dispatch(Events.E_LocalConfigLoadSuccess);
            appDataModel.SetBooks(booksParseBuffer.Values);
            appDataModel.SetAnimations(animationsParseBuffer.Values);
        }
        else
        {
            dispatcher.Dispatch(Events.E_LocalConfigLoadFailed);
        }

        LoadRemoteConfigs();


    }

    private void LoadLocalConfigs()
    {
        if (!File.Exists(configPath))
        {
```

```
            localConfigLoaded = false;
            return;
        }

        string jsonString = File.ReadAllText(configPath);
        if (string.IsNullOrEmpty(jsonString))
        {
            localConfigLoaded = false;
            return;
        }

        localConfigLoaded = ParseConfigs(jsonString);

    }

    private void LoadRemoteConfigs()
    {
        JSONRequestModel request = new
JSONRequestModel(NetworkRequests.GetJsonData, OnRemoteLoadSuccess, errorHandler:
OnNetworkError);
        dispatcher.Dispatch(Events.E_DownloadJSON, request);
    }

    private List<CollectionModel> SelectBooksToUpdate(Dictionary<int,
CollectionModel> localBooks, Dictionary<int, CollectionModel> remoteBooks)
    {
        List<CollectionModel> toUpdate = new List<CollectionModel>();
        foreach (var remoteBook in remoteBooks)
        {
            if (!localBooks.ContainsKey(remoteBook.Key))
            {
                Debug.Log("Need update book " + remoteBook.Key);
                remoteBook.Value.ClearAllLocalFiles();
                toUpdate.Add(remoteBook.Value);
                continue;
            }
            CollectionModel localBook = localBooks[remoteBook.Key];
            if (localBook.Version != remoteBook.Value.Version)
            {
                Debug.Log("Need update book " + remoteBook.Key);
                remoteBook.Value.ClearAllLocalFiles();
                toUpdate.Add(remoteBook.Value);
                continue;
            }

            if (!remoteBook.Value.DataSetDatExist || !remoteBook.Value.ImageExist
|| !remoteBook.Value.DataSetXMLExist)
            {
                toUpdate.Add(remoteBook.Value);
                continue;
            }
        }
        return toUpdate;
```

```
    }

    private List<ARContentModel> SelectAnimationsToUpdate(Dictionary<int,
ARContentModel> localAnimations, Dictionary<int, ARContentModel> remoteAnimations)
    {
        List<ARContentModel> toUpdate = new List<ARContentModel>();
        foreach (var remoteAnimation in remoteAnimations)
        {
            if (!localAnimations.ContainsKey(remoteAnimation.Key))
            {
                Debug.Log("Need update book " + remoteAnimation.Key);
                remoteAnimation.Value.ClearAllLocalFiles();
                toUpdate.Add(remoteAnimation.Value);
                continue;
            }
            ARContentModel localAnimation = localAnimations[remoteAnimation.Key];
            if (localAnimation.Version != remoteAnimation.Value.Version)
            {
                Debug.Log("Need update book " + remoteAnimation.Key);
                remoteAnimation.Value.ClearAllLocalFiles();
                toUpdate.Add(remoteAnimation.Value);
            }
        }
        return toUpdate;
    }


    public const string COLLECTIONS_KEY = "collection";
    public const string ANIMATIONS_KEY = "arcontent";

    private bool ParseConfigs(string jsonString)
    {
        if (string.IsNullOrEmpty(jsonString))
        {
            return false;
        }
        JSONNode json = JSONCreator.Parse(jsonString);

        JSONArray books = json[COLLECTIONS_KEY].AsArray;
        JSONArray animations = json[ANIMATIONS_KEY].AsArray;

        booksParseBuffer.Clear();
        foreach (JSONNode bookJSON in books)
        {

            var book = new CollectionModel(bookJSON);
            booksParseBuffer.Add(book.ID, book);
            Debug.Log(book.ID);
        }

        animationsParseBuffer.Clear();
        foreach (JSONNode animationJSON in animations)
        {
```

```
            var animation = new ARContentModel(animationJSON);
            if (!booksParseBuffer.ContainsKey(animation.CollectionID))
            {
                Debug.LogError("Cant find book for animation: " + animation.ID);
                continue;
            }
            var book = booksParseBuffer[animation.CollectionID];
            animation.SetCollection(book);
            book.AddAnimation(animation);
            animationsParseBuffer.Add(animation.ID, animation);
        }
        return booksParseBuffer.Count > 0 && animationsParseBuffer.Count > 0;
    }

    private void OnNetworkError(string error)
    {
        remoteConfigLoaded = false;
        if (!localConfigLoaded)
        {
            // load again
            dispatcher.Dispatch(Events.E_RemoteConfigLoadFailed);
            dispatcher.AddListener(Events.E_RetryDownloading, Execute);
        }
        else
        {
            dispatcher.Dispatch(Events.E_StartUsingLocalConfigs);
        }
    }

    private void OnRemoteLoadSuccess(string jsonString)
    {
        remoteConfigLoaded = ParseConfigs(jsonString);
        if (File.Exists(configPath))
        {
            File.Delete(configPath);
        }
        File.WriteAllText(configPath, jsonString);


        if (remoteConfigLoaded)
        {
            dispatcher.Dispatch(Events.E_RemoteConfigLoadSuccess);
        }
        else
        {
            dispatcher.Dispatch(Events.E_RemoteConfigLoadFailed);
            if (!localConfigLoaded)
            {
                // load again
                dispatcher.AddListener(Events.E_RetryDownloading, Execute);
            }
            return;
        }
```

```
        if (remoteConfigLoaded)
        {

            if (localConfigLoaded)
            {
                booksToUpdate = SelectBooksToUpdate(appDataModel.Books,
booksParseBuffer);
                animationsToUpdate =
SelectAnimationsToUpdate(appDataModel.Animations, animationsParseBuffer);
            }
            else
            {
                booksToUpdate = appDataModel.Books.Values;
                animationsToUpdate = appDataModel.Animations.Values;
            }

            appDataModel.SetBooks(booksParseBuffer.Values);
            appDataModel.SetAnimations(animationsParseBuffer.Values);


            dispatcher.Dispatch(Events.E_UpdateBooksData, booksToUpdate);
        }

    }

}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.IO;

public class UpdateBooksCommand : BaseCommand
{

    [Inject] public INetworkService networkService { get; set; }

    private Queue<FileRequestModel> requestQueue = new Queue<FileRequestModel>();

    private List<FileRequestModel> executingRequests = new
List<FileRequestModel>();

    private int paralelRequestsCount = 10;

    private int successRequests = 0;

    private int totalRequests = 0;

    private Coroutine progressWorker = null;

    public override void Execute()
    {
```

```
        var toUpdate = eventData.data as IEnumerable<CollectionModel>;

        requestQueue = new Queue<FileRequestModel>();
        executingRequests = new List<FileRequestModel>();
        successRequests = 0;
        totalRequests = 0;
        progressWorker = null;

        foreach (var book in toUpdate)
        {
            if (!Directory.Exists(book.CacheDirectoryPath))
            {
                Directory.CreateDirectory(book.CacheDirectoryPath);
            }
            if (!book.DataSetDatExist)
            {
                FileRequestModel request = new
FileRequestModel(book.DataSetDatUrl, (req) => SaveDat(req, book.DataSetDatPath),
errorHandler: OnError, priority: ThreadPriority.High);
                requestQueue.Enqueue(request);
            }
            if (!book.DataSetXMLExist)
            {
                FileRequestModel request = new
FileRequestModel(book.DataSetXMLUrl, (req) => SaveXML(req, book.DataSetXMLPath),
errorHandler: OnError, priority: ThreadPriority.High);
                requestQueue.Enqueue(request);
            }
            if (!book.ImageExist)
            {
                FileRequestModel request = new FileRequestModel(book.ImageUrl,
(req) => SaveImage(req, book.ImagePath), errorHandler: OnError, priority:
ThreadPriority.High);
                requestQueue.Enqueue(request);
            }
        }

        totalRequests = requestQueue.Count;
        paralelRequestsCount = Mathf.Min(paralelRequestsCount, totalRequests);

        if (totalRequests <= 0)
        {
            dispatcher.Dispatch(Events.E_UpdateBooksDataSuccess);
            return;
        }

        for (int i = 0; i < paralelRequestsCount; i++)
        {
            var request = requestQueue.Dequeue();
            executingRequests.Add(request);
            dispatcher.Dispatch(Events.E_DownloadFile, request);
        }
```

```
        progressWorker = executor.Execute(CalculateTotalProgress());

    }

    private void SaveImage(FileRequestModel request, string path)
    {
        OnRequestCompleted(request);
        if (File.Exists(path))
        {
            File.Delete(path);
        }
        File.WriteAllBytes(path, request.rawFile);

    }

    private void SaveDat(FileRequestModel request, string path)
    {
        OnRequestCompleted(request);
        if (File.Exists(path))
        {
            File.Delete(path);
        }
        File.WriteAllBytes(path, request.rawFile);
    }

    private void SaveXML(FileRequestModel request, string path)
    {
        OnRequestCompleted(request);
        if (File.Exists(path))
        {
            File.Delete(path);
        }
        File.WriteAllBytes(path, request.rawFile);
    }

    private void OnRequestCompleted(FileRequestModel request)
    {
        successRequests++;
        executingRequests.Remove(request);
        if (requestQueue.Count > 0)
        {
            var newRequest = requestQueue.Dequeue();
            executingRequests.Add(newRequest);
            dispatcher.Dispatch(Events.E_DownloadFile, newRequest);
        }

        if (executingRequests.Count <= 0 || successRequests == totalRequests)
        {
            Debug.Log("Log executingRequests on success: " +
executingRequests.Count);
            OnLoadSuccess();
        }
    }
```

```
    private void OnLoadSuccess()
    {
        if (dispatcher.HasListener(Events.E_RetryDownloading, Execute))
        {
            dispatcher.RemoveListener(Events.E_RetryDownloading, Execute);
        }
        dispatcher.Dispatch(Events.E_UpdateBooksDataSuccess);
    }

    private void OnError(string error)
    {
        executor.StopExecution(progressWorker);
        networkService.AbortAllRequests();
        dispatcher.Dispatch(Events.E_UpdateBooksDataFail);

        if (!dispatcher.HasListener(Events.E_RetryDownloading, Execute))
        {
            dispatcher.AddListener(Events.E_RetryDownloading, Execute);
        }
    }

    private IEnumerator CalculateTotalProgress()
    {
        while (true)
        {
            if (executingRequests.Count <= 0 || totalRequests == successRequests)
            {
                yield break;
            }

            float totalProgress = successRequests;
            foreach (var request in executingRequests)
            {
                totalProgress += request.Progress;
            }
            dispatcher.Dispatch(Events.E_UpdateBooksDataProgress, totalProgress /
totalRequests);
            Debug.Log("E_UpdateBooksDataProgress: " + totalProgress /
totalRequests);
            yield return null;
        }
    }

}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class UIMapShowCommand : BaseCommand
{
```

```
    private const string NAME = "UIMap";
    private const string PATH = "Prefabs/UI/UIMap";

    public override void Execute()
    {
        var prefab = Resources.Load<GameObject>(PATH);
        if (prefab == null)
        {
            Debug.LogError("prefab is null at " + PATH);
            Fail();
        }

        var GO = GameObject.Instantiate<GameObject>(prefab);
        GO.name = NAME;

        var view = GO.GetComponent<UIMapView>();

        if (view == null)
        {
            Debug.LogError("GO has no view at " + PATH);
            Fail();
        }

    }

}


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

public class AddAllBooksMarkersCommand : BaseCommand
{

    [Inject] public AppDataModel appData { get; private set; }

    public override void Execute()
    {
        Retain();
        executor.Execute(AddMarkers());
    }

    private IEnumerator AddMarkers()
    {

        var books = appData.Books.Values;
        if (books == null)
        {
            Fail();
            yield break;
        }
```

```
        yield return new WaitForSeconds(1f);
        TrackerManager.Instance.InitTracker<ObjectTracker>();
        TrackerManager.Instance.GetTracker<ObjectTracker>().Start();
        ObjectTracker objTracker =
TrackerManager.Instance.GetTracker<ObjectTracker>();

        GameObject arMarkers = GameObject.Find("MarkerTargets");

        objTracker.Stop();
        yield return new WaitForSeconds(1f);
        foreach (var book in books)
        {
            if (!book.CheckAllAnimationsPresent())
            {
                continue;
            }
            DataSet dataSet = objTracker.CreateDataSet();

            if (book.DataSetXMLExist)
            {
                if (dataSet.Load(book.DataSetXMLPath,
VuforiaUnity.StorageType.STORAGE_ABSOLUTE))
                {

                    IEnumerable<TrackableBehaviour> tbs =
TrackerManager.Instance.GetStateManager().GetTrackableBehaviours();
                    foreach (TrackableBehaviour tb in tbs)
                    {

                        if (tb.name == "New Game Object")
                        {
                            var imageTarget = tb as ImageTargetBehaviour;

                            if (imageTarget != null)
                            {
                                float aspectRatio = imageTarget.GetSize().x /
imageTarget.GetSize().y;
                                float sizeMarker = 1f;
                                imageTarget.SetHeight(sizeMarker);
                                imageTarget.SetWidth(sizeMarker * aspectRatio);
                            }
                            var animation = book.Animations.Find((x) => x.Name ==
tb.TrackableName);
                            tb.gameObject.name = animation == null ? "" :
animation.ID.ToString() + "(" + tb.TrackableName + ")";
                            var eventHandler =
tb.gameObject.AddComponent<StrangeIOCTrackableEventHandler>();
                            eventHandler.ImageTargetID = animation == null ? 0 :
animation.ID;
                            eventHandler.dispatcher = dispatcher;

                            //eventHandler.CostumeSize = tb.
```

```
                            if (arMarkers != null)
                                tb.transform.parent = arMarkers.transform;

                            tb.transform.localScale = Vector3.one;
                            tb.transform.localPosition = Vector3.zero;
                        }
                    }
                    if (dataSet == null)
                    {
                        Debug.LogError("dataset is null");
                    }
                    book.SetDataSet(dataSet);
                    objTracker.ActivateDataSet(dataSet);
                }
                else
                {
                    Debug.LogError("cant load dataset");

                }

            }
            else
            {
                Debug.LogError("dataset not exist");

            }
        }
        yield return new WaitForSeconds(1f);
        objTracker.Start();
        dispatcher.Dispatch(Events.E_MarkersAdded);
        Release();
    }

}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using SimpleJSON;
using System.IO;

public class ARContentModel
{
    public const string COLLECTION_ID_KEY = "collection";
    public const string BUNDLE_ANDROID_URL_KEY = "bundle_android";
    public const string BUNDLE_IOS_URL_KEY = "bundle_ios";
    public const string ID_KEY = "id";
    public const string VERSION_KEY = "version";
    public const string NAME_KEY = "name";
    public const string SCALE_KEY = "scale";
```

```
    public string BundlePath { get { return Path.Combine(CacheDirectoryPath,
ID.ToString()); } }
    public string BundleUrl { get; private set; }
    public int Version { get; private set; }
    public int ID { get; private set; }
    public int CollectionID { get; private set; }
    public float Scale { get; private set; }
    public string Name { get; private set; }
    public CollectionModel Collection { get; private set; }
    public string CacheDirectoryPath {  get { return Collection != null ?
Path.Combine(Collection.CacheDirectoryPath, ID.ToString()) : null; } }
    public bool BundleExist { get { return File.Exists(BundlePath); } }

    public ARContentModel(JSONNode json)
    {
#if UNITY_ANDROID || UNITY_EDITOR
        BundleUrl = NetworkRequests.ServerUrl + json[BUNDLE_ANDROID_URL_KEY];
#elif UNITY_IOS
        BundleUrl = NetworkRequests.ServerUrl + json[BUNDLE_IOS_URL_KEY];
#endif
        Version = json[VERSION_KEY].AsInt;
        CollectionID = json[COLLECTION_ID_KEY].AsInt;
        ID = json[ID_KEY].AsInt;
        Scale = json[SCALE_KEY].AsFloat;
        Name = json[NAME_KEY];
    }

    public void SetCollection(CollectionModel book)
    {
        Collection = book;
    }

    public void ClearAllLocalFiles()
    {
        if (BundleExist)
        {
            File.Delete(BundlePath);
        }
    }

}
```

Додаток 3
Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

Програмне забезпечення для розширення експозиції музею на основі технології доповненої реальності

Виконав: Саленко Антон Володимирович

Керівник: к.т.н, доцент Сулема Євгенія Станіславівна

Київ – 2019

ПОСТАНОВКА ЗАДАЧІ

Мета проекту: Створення мобільного додатку для відвідувачів музею, який виконує роль мультимедійного гіда та надає змогу переглядати анімаційний продукт у доповненій реальності призначений для зацікавлення відвідувачів.

ПОСТАНОВКА ЗАДАЧІ

Завдання:

1. Проаналізувати та обрати технологію доповненої реальності
2. Розробити продукт на основі технології доповненої реальності базуючись на методах відстеження зображення
3. Протестувати додаток на реальних зображеннях і об’єктах

3

АКТУАЛЬНІСТЬ

- Технологія доповненої реальності уже починає поширюватися у популярних музеях світу
- В Україні музеї не мають віртуальних експозицій щоб змагатися з конкурентами за кордоном
- Додаткова можливість зацікавити підростаюче покоління за допомогою інновацій та заохотити до відвідування музеїв

ОБРАНІ ЗАСОБИ ВИРІШЕННЯ ПРОБЛЕМИ

- вбудовані технології ARKit, ARCore
- технологія відстежування зображень
- проста імплементація на мобільних платформах

- зручний для реалізації прототипів
- кросплатформність
- можливість глибокого налаштування графіки

ШАБЛОНИ ПРОЕКТУВАННЯ

- MVCS, або Model-View-Controller-Service - основний архітектурний шаблон проекту
- ECS, або Entity Component System - шаблон проектування що використовується у Unity
- Dependency Injection - покращений шаблон Singleton
- Dispatcher - відповідає за виклик подій з будь-якої точки програми
- Command - виконує роль контроллера та являється ядром системи
- Mediator - виконує роль зв’язку між MVCS та ECS шаблонами
- Context - описує стратегію виконання програми
- Binder - зв’язує події з командами які їх обробляють

КОД АЛГОРИТМУ ОРІЄНТАЦІЇ ОБ'ЄКТУ ЗА ДОПОМОГОЮ ГІРОСКОПУ

Мета реалізації: при втраті маркера з поля зору камери створити імітацію доповненої реальності орієнтуючи об'єкт вертикально

```
var gyroRotation = Input.gyro.attitude;
transform.LookAt(targetTransform);
var rotation = Quaternion.Euler(-90f, 0f, 0f) * gyroRotation;
transform.localRotation = Quaternion.Euler(0f, 180f, 0f) *
    Quaternion.Euler(0f, 0f, rotation.eulerAngles.z) *
    Quaternion.Euler(-rotation.eulerAngles.x, 0f, 0f) *
    transform.localRotation;
```

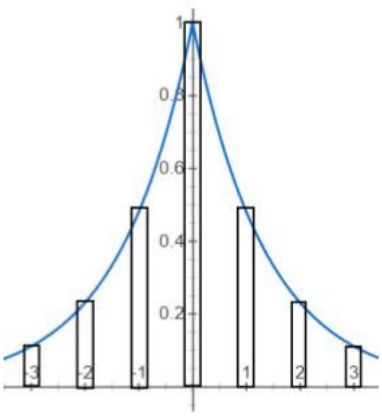
АЛГОРИТМ РОЗМИТТЯ ЗАДНЬОГО ФОНУ

Порядок виконання: наприкінці рендеру сцени отримати рендер текстуру. Для кожного пікселя виконати наступні обчислення

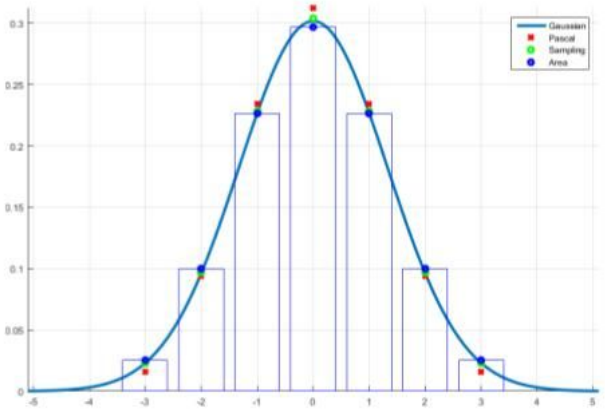
$$x_0' = \frac{\left(p^3 \cdot x_3 + p^2 \cdot x_2 + p^1 \cdot x_1 + x_0 + p \cdot x_{-1} + p^2 \cdot x_{-2} + p^3 \cdot x_{-3}\right)}{2 \cdot p \cdot (1 + p \cdot (1 + p)) + 1}$$

Обчислення виконуються чотири рази, перший раз беручи за сусідні значення по горизонталі, другий по вертикалі, а наступні два повторити для кращого ефекту.

ПОРІВНЯННЯ РЕЗУЛЬТАТІВ РОБОТИ АЛГОРИТМУ



Запропонований алгоритм



Розмиття за Гуассом

КОМПОНЕНТ ІНТЕРАКТИВНА МАПА

Мета - зробити зручним орієнтацію користувача по музею та позначення місцезнаходження маркерів на карті

Опис роботи:

- доступ до мапи з меню камери
- меню у вигляді спливаючого вікна
- підвантаження зображення мапи та позицій маркерів з ресурсів або мережі інтернет
- управління жестами, а саме перенесення та зміна розміру
- зміна кольору позначки на мапі в залежності від того чи зв' язаний маркер було просканово

МОДУЛЬ ЗАВАНТАЖЕННЯ РЕСУРСІВ

Основні переваги:

- Немає необхідності оновлення версії ПЗ з кожним новим анімаційним матеріалом або його коригуванням
- Зменшення розміру ПЗ
- Швидке тестування нового матеріалу на мобільних девайсах без необхідності створення нової версії ПЗ
- Можливість швидко виправити несправні ресурси

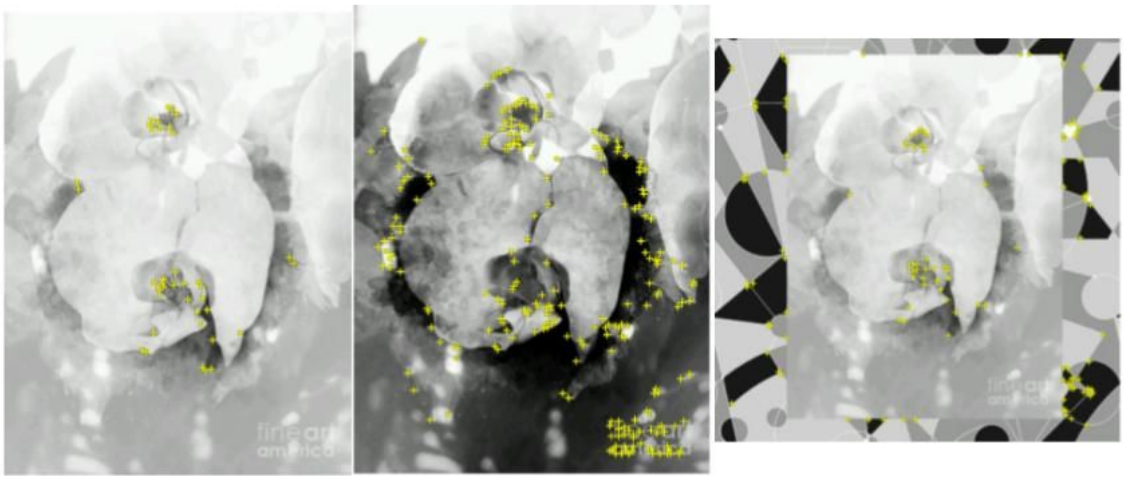
КРИТЕРІЇ ОЦІНЮВАННЯ МАРКЕРІВ

- Маркер має бути достатньо контрастним з різкими переходами кольорів
- Відсутність повторень об’єктів на маркері
- Відсутність спільних об’єктів на різних маркерах
- Кількість контурних вузлів

СПОСОБИ ПОКРАЩЕННЯ МАРКЕРІВ

- Підвищення контрастності
- Гамма-корекція
- Збільшення кількості контурних вузлових точок шляхом додавання геометричних фрагментів

ПРИКЛАД ПОКРАЩЕННЯ МАРКЕРУ



КРИТЕРІЇ ОЦІНЮВАННЯ ЯКОСТІ РОЗРОБЛЕНОГО ПЗ

Критерій	Цільові значення	Реальні значення
частка площі сканера яку займає маркер при його миттєвому розпізнаванні	35%	40%
частка ефективної площі маркера яка потрапляє у сканер за якої він розпізнається	60%	50%
швидкість завантаження ресурсів (50Мб)	менше 10с	12с

ВИСНОВКИ

1. Проаналізовано програмні рішення, що використовують доповнену реальність
2. Застосований архітектурний шаблон MVCS
3. Розроблено алгоритм орієнтації об’єкту за допомогою гіроскопу
4. Розроблено алгоритм розмивання заднього фону
5. Розроблено зручний користувацький інтерфейс
6. Розроблено інтерактивну мапу та реалізовано управління жестами
7. Загальний висновок - ПЗ готове до випуску

Дякую за увагу!

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗШИРЕННЯ ЕКСПОЗИЦІЇ МУЗЕЮ НА ОСНОВІ ТЕХНОЛОГІЇ ДОПОВНЕНОЇ РЕАЛЬНОСТІ

Програма та методика тестування

ДП.045480-04-51

"ПОГОДЖЕНО"

Керівник проекту:

_____ Є.С. Сулема

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ А.В. Саленко

2018

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Програмне забезпечення для розширення експозиції музею на основі технології доповненої реальності являє собою мобільний додаток, створений з використанням технологій Unity та Vuforia.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1) коректність та швидкість завантаження ресурсів;
2) тестування UI;
3) тестування якості маркерів;
4) тестування швидкості сканування та стабільності відстеження;
5) відповідність вимогам технічного завдання.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом Gray Box Testing. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам. Тестування відбувається на рівні «системного тестування».

Використовуються наступні методи:

1) функціональне тестування на базі вимог та на базі тестових випадків;
2) димове тестування (Smoke testing);
3) тестування інтерфейсу.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується засобами інструментарію Unity Profiler, Android Debug Bridge.

Працездатність програмного забезпечення перевіряється шляхом:

1) динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати;
2) динамічного ручного тестування на відповідність функціональним вимогам;
3) статичного тестування коду;
4) тестування верстки;
5) тестування зручності використання;
6) тестування інтерфейсу;
7) тестування стабільності роботи при різних умовах;
8) тестування швидкодії та розміру використовуваних ресурсів.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗШИРЕННЯ ЕКСПОЗИЦІЇ МУЗЕЮ НА ОСНОВІ ТЕХНОЛОГІЇ ДОПОВНЕНОЇ РЕАЛЬНОСТІ

Керівництво користувача

ДП.045480-05-34

"ПОГОДЖЕНО"

Керівник проекту:

_____ Є.С. Сулема

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ А.В. Саленко

2019

ЗМІСТ

1. Опис структури програмного забезпечення

Програмне забезпечення для розширення експозиції музею на основі технології доповненої реальності складається з набору модулів. Основне призначення – розширити експозицію музею віртуальними експозиціями. Структура програмного забезпечення складається з таких модулів:

- модуль завантаження ресурсів з віддаленого серверу;
- модуль користувацького інтерфейсу;
- модуль динамічного завантаження маркерів;
- модуль роботи з доповненою реальністю.

2. Опис роботи програмного забезпечення

Головне меню (рис. 1) складається з трьох кнопок:

- кнопка для переходу в режим камери;
- кнопка для відкривання списку колекцій;
- кнопка для відкривання меню налаштувань.

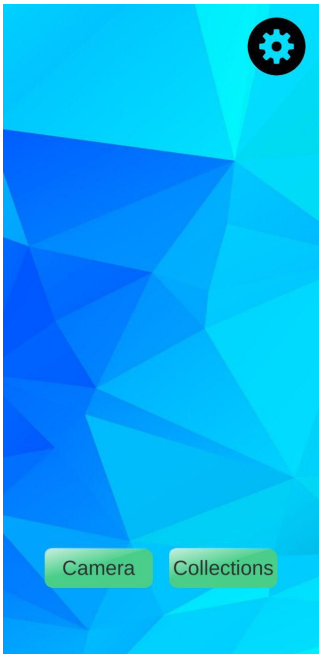


Рис. 1. Головне меню

Для початку користувачу необхідно зайти в меню колекцій (рис. 2) де він може завантажити відповідну до музею колекцію.

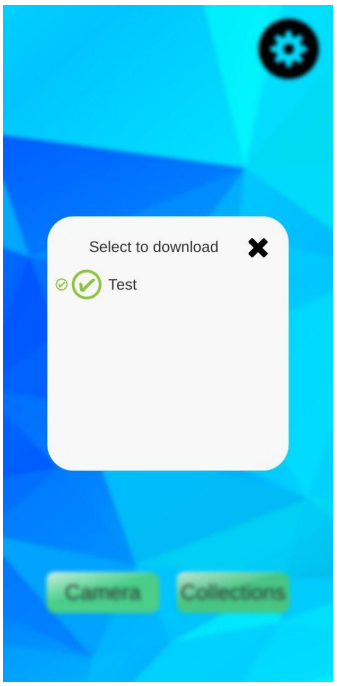


Рис. 2. Меню перегляду колекцій для завантаження

Після завантаження колекції користувач може перейти у меню камери і відсканувати маркер. Для цього необхідно навести сканер на камеру і дочекатися поки зображення стане якісним.

Рис. 3. Меню камери з віртуальною експозицією

У меню камери (рис. 3) є 5 кнопок:

- кнопка для видалення віртуальної експозиції зі сцени;
- кнопка для переходу у меню мапи музею;
- кнопка для фотозйомки;
- кнопка для зупинки анімації віртуальної експозиції;
- кнопка для повернення у головне меню.

Кнопка для фотозйомки робить фото і миттєво відкриває стандартне меню Android “поділитися”. Так користувач має змогу надіслати фото у соціальні мережі чи на пошту, або зберегти його.

Якщо втратити маркер з поля зору сканера то віртуальна експозиція буде демонструватися по центрі камери.

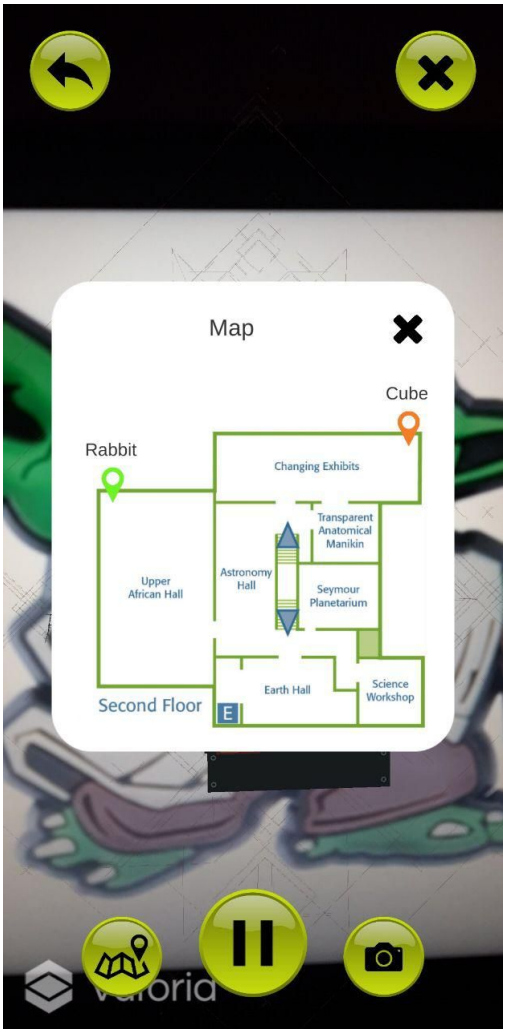


Рис. 4. Меню мапи музею

У меню мапи (рис. 4) користувач має змогу її масштабувати і переглядати місця в якому є віртуальні експозиції, їх назву та інформацію про те чи користувач їх уже відсканував.

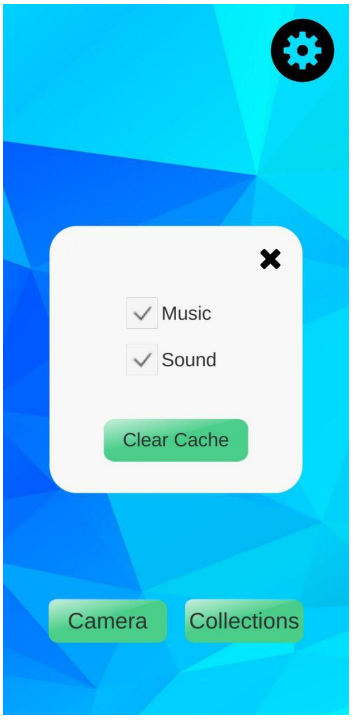


Рис. 5. Меню налаштувань

У меню налаштувань (рис. 5) користувач може увімкнути чи вимкнути звук користувацького інтерфейсу, звук віртуальних експозицій та видалити ресурси програми якщо є така потреба, наприклад якщо вони відображаються не корректно.