Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра програмних систем і технологій

УДК 004.942

На правах рукопису

ВИПУСКНА КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Тема: "Додаток для пошуку розваг з використанням Unity та доповненої реальності"

(назва згідно з наказом ректора)

Спеціальність – 121 "Інженерія програмного забезпечення"

ПОЯСНЮВАЛЬНА ЗАПИСКА

БР.ІПЗ - 22.00.00.000

(позначення)

Студент

ІПЗ-42 **(Владислав ФЕСЮК)**

(шифр групи) (підпис) (дата)(розшифровка підпису)

Науковий керівник

ас. **(Єлизавета ЖАБСЬКА)**

(посада) (підпис) (дата) (розшифровка підпису)

Консультант з питань нормконтролю

Фахівець **(Тамара ЧАПОВСЬКА)**

Допускається до захисту

Завідувач кафедри

д.т.н., проф. **(Олексій БИЧКОВ)**

Київ - 2021

Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра програмних систем і технологій

Освітньо-кваліфікаційний рівень бакалавр

Спеціальність 121 “Інженерія програмного забезпечення”

ЗАТВЕРДЖЕНО

Зав. кафедри _програмних систем і технологій__

_____ (Олексій БИЧКОВ)

(підпис) (прізвище та ініціали)

ЗАВДАННЯ

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Фесюку Владиславу Валерійовичу

(прізвище, ім’я, по-батькові)

1. Тема бакалаврської роботи “ Додаток для пошуку розваг з використанням Unity та доповненої реальності ”

керівник проекту (роботи) Жабська Єлизавета Олегівна

затверджені наказом вищого навчального закладу від “ 07 ” грудня 2020 р. № ___

2. Строк подання студентом роботи 3 червня 2021 р.

3. Вихідні дані до проекту (роботи) проблема безкорисного витрачання часу

4. Зміст розрахунково - пояснювальної записки(перелік питань, які потрібно розробити)

1. Дослідження проблематики поставленої задачі

2. Проаналізувати варіанти вирішення проблеми

3. Обрати оптимальний варіант реалізації програмного забезпечення

4. Реалізувати продукт

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

1. Діаграми використання (додоток Д)
2. Діаграми послідовності (додаток Д)
3. Діаграми пакетів (додаток Д)
4. Інтерфейс користувача (рис 4.1 ст. 47)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
		Жабська Елизавета Олегівна	Фесюк Владислав Валерійович

7. **Дата видачі завдання** 10 грудня 2020 р.

Керівник _____ (Элизавета ЖАБСЬКА)

Завдання прийняв до виконання ____ _____ (Владислав ФЕСЮК)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Розбір проблематики проекту	25.01.2021	виконано
2	Знаходження наукових матеріалів	05.02.2021	виконано
3	Аналіз вирішення проблеми	17.02.2021	виконано
4	Розробка алгоритму роботи додатку	22.03.2021	виконано
5	Вибір засобів реалізації проекту	01.04.2021	виконано
6	Документування додатку	10.04.2021	виконано
7	Програмна реалізація додатку	25.05.2021	виконано
8	Затвердження пояснювальної записки роботи завідувачем кафедри	30.05.2021	виконано

Студент – бакалавр _____(Владислав ФЕСЮК)

Керівник роботи _____(Єлизавета ЖАБСЬКА)

АНОТАЦІЯ

Випускна кваліфікаційна бакалаврська робота: 74 с., 19 рис., 5 додат.

Тема: додаток для пошуку розваг з використанням Unity та доповненої реальності.

Об'єкт дослідження: аналітичні дані щодо проведення людиною вільного часу та складність організації власних заходів.

Мета роботи: розробити програмне забезпечення, що вирішувало б проблему вибору людиною способу проведення вільного часу та організації власних заходів.

Предмет дослідження: технології доповненої реальності у поєднанні з інтерактивною картою та Unity.

Результати дослідження:

Досліджено можливості застосування доповненої реальності та Unity у сфері розвитку особистості. Створено прототип нового продукту у сфері пошуку та організації заходів. Отримана реалізація програмного продукту зможе допомогти користувачу у виборі варіантів проведення свого вільного часу та поширенні інформації про свої власні заходи.

Висновок:

В результаті роботи було розроблено програмний продукт, що міг би вирішити досліджені проблеми та зміг би покращити людську життєдіяльність.

АННОТАЦИЯ

Выпускная квалификационная бакалаврская работа: 74 с., 19 рис., 5 додат.

Тема: приложение для поиска развлечений с использованием Unity и дополненной реальности.

Объект исследования: проведение человеком свободного времени и сложность организации собственных мероприятий.

Цель работы: разработать программное обеспечение, решало бы проблему выбора человеком способа проведения свободного времени и организации собственных мероприятий.

Предмет исследования: сфера жизни человека, связана с его социальными потребностями.

Результаты исследования: исследованы возможности применения дополненной реальности и Unity в сфере развития личности. Создан прототип нового продукта в области поиска и организации мероприятий. Полученная реализация программного продукта сможет помочь пользователю в выборе вариантов проведения своего свободного времени и распространении информации о своих собственных мероприятиях.

Вывод:

В результате работы был разработан программный продукт, который мог бы решить исследованы проблемы и смог бы улучшить человеческую жизнедеятельность.

ANNOTATION

Final qualifying Bachelor's work: 74 pages, 19 pictures, 5 adds.

Topic: Entertainment search application using Unity and augmented reality.

Object of research: the person's spending of free time and the complexity of organizing their own events.

Purpose of the work: develop software that would solve the problem of a person's choice of a way to spend their free time and organize their own events**.**

Subject of the study: the sphere of a person's life is connected with his social needs.

Results of the study: it has been found the possibilities of using augmented reality and Unity in the field of personality development. A prototype of a new product in the field of search and organization of events was created. The resulting implementation of the software product will be able to help users in choosing options for spending their free time and disseminating information about their own events.

Conclusion: as a result of the work, a software product was developed that could solve the investigated problems and could improve human life.

ЗМІСТ

Стр.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД - база даних

ІС - інформаційна система

ООП - об’єктно-орієнтоване програмування

ОС - операційна система

ПП - програмний продукт

ПЗ - програмне забезпечення

СУБД - система управління базами даних

ВСТУП

Актуальність роботи

Більшість людей переконані, що найважливішими у житті є проекти, робота, навчання, пригоди, досягнення та інше. Коли певна сфера не розвивається так швидко, як хотілось би, коли мета не досягається необхідними інструментами, то у багатьох настає так зване "вигорання". Люди роками живуть на гойдалках, де завзята праця змінюється на вигорання - і так по колу.

Насправді всі проекти і життєві плани будуються на однаковій для всіх основі. Це фізика, тіло, режим, сон, ресурс, безпека та відпочинок. Отже, якість організації життя суспільства і людини напряму залежить від цих чинників, зокрема від відпочинка. Резерви енергії кожного з членів суспільства мають здатність вичерпуватися, а для їх відновлення вкрай потрібен відпочинок. На нього, як правило, виділяють так званий “вільний” час. Саме дозвілля як частина вільного часу, надає можливість не тільки відновити запаси енергії, а дозволяє поєднувати в ньому різні види діяльності, які позитивно впливають на інтелектуальну, творчу, фізичну і виробничу сфери людської життєдіяльності. Це дає можливість поєднувати корисне з приємним. Якісно сформоване дозвілля - частина життя, яку весь час можна вдосконалювати. Дозвілля надає людині ресурси і, за правильного використання, покращує загальний рівень самопочуття. Завдяки ньому, сучасна молода людина може розвивати багато сторін своєї особистості, у тому числі власний талант. Але для цього необхідно мати правильний та серйозний підхід до вибору відпочинку, зокрема усвідомлювати його необхідність та вплив на загальне самопочуття та власний добробут. Усвідомлення цього допоможе потім обирати правильний та корисний спосіб проведення дозвілля.

Для більшості людей різного віку, саме дозвілля є головним джерелом здобуття нових знайомств, комунікації та часом для хобі, самореалізації. На жаль, раціональне

використання дозвілля задля виконання життєвої мети та самореалізації ще не стали звичкою та загальноприйнятою позицією у сучасних реаліях.

Вільний час українська молодь проводить здебільшого за так званим «пасивним дозвіллям» або на природі. Популярними є й розважальні види дозвілля, які потребують матеріальних витрат: клуби, кафе, кінотеатри, аквапарки тощо. Третина присвячує вільні хвилини своєму хобі. На вибір виду дозвілля впливають рівень освіти і матеріального благополуччя, тип поселення і регіон проживання, стать і вид зайнятості. Наприклад, жителі сільських регіонів частіше проводять час на дискотеках, адже там не такий великий вибір розваг, до того ж - інший середній прибуток, а отже, інший бюджет для організації відпочинку.

Середній дохід та матеріальне забезпечення прямо впливають на вибір способу проведення вільного часу. Найнижчою популярністю у молоді користуються державні культосвітні заклади, бібліотеки, хоча вони, як правило, є безкоштовними.

Поряд із великим вибором серед видів та способів відпочинку існує велика кількість недоліків та проблем, які виникають саме у сфері дозвілля. Найголовніша - розповсюдженість відсутності культури дозвілля серед молоді.

Щоб вирішити такі проблеми, потрібно їх знати. Отже, вивчення питань, пов'язаних з суперечностями сфери дозвілля, допоможе розвинути інтерес людей до теми використання вільного часу. Крім того, це допоможе створювати актуальні для види відпочинку, які не тільки викличуть інтерес, а ще й нададуть поштовх для культурного розвитку й розвитку духовних потреб. Отже, проблема є актуальною і її необхідно вирішувати.

Таким чином, крім вивчення проблем дозвілля, необхідно займатися привиттям молоді нових цінностей. Це можливо робити тільки за наявності альтернативного відпочинку, який зацікавить і наочно покаже свою користь.

Живе спілкування відіграє важливу роль в розвитку та житті людини. Не можна уявити всебічно розвинену особистість, яка весь час сидить вдома та спілкується тільки в соціальних мережах. Адже кожен потрапляв у ситуацію, коли хотів би

погуляти з друзями, але навіть не знає, куди б сходити. Дана ситуація типова для сучасного покоління, адже ми звикли, що вибір роблять за нас або максимально його спрощують. Більшість програм працюють за алгоритмами, підбираючи найцікавіше. Вибір - лиши натиснути на потрібну клавішу. Тому пошук цікавих заходів навколо часто витісняється переглядом чергового ролику, адже не достатньо просто відкрити карту та подивитися на різні сквери, кіно, театри.

Ця проблема є основною у даній роботі. Це визначає її актуальність у різних аспектах життєдіяльності нашого часу.

Порівняння роботи з відомими розв'язаннями проблеми

На даний момент проблема витрати вільного часу вирішується по-різному. До таких методів можна віднести платформи для самонавчання з відеоуроками, сервіси для перегляду відео та фільмів, соціальні мережі, електронні книжки. Кожен з цих варіантів проводження вільного часу може мати як позитивний вплив на людину так і ні.

Проблема, що вирішується - це розвиток особистості, а особливо молодого покоління, економія часу користувачів, створення умов для більш корисного та цікавого проводження вільного часу, організації власних заходів та залучення нової аудиторії.

Мета і задачі роботи

Метою бакалаврської роботи є розроблення програмного забезпечення, що допоможе людині з пошуком варіантів проведення свого вільного часу та створенням власних заходів.

Об’єктом дослідження є аналітичні дані щодо проведення людиною вільного часу та складність організації власних заходів

Предметом дослідження є технології доповненої реальності у поєднанні з інтерактивною картою та Unity.

Методи дослідження

Для досягнення мети були використані методи з використання доповненої реальності, алгоритми аналізу положення об’єктів у просторі, та побудови їх.

Наукова новизна отриманих результатів

Досліджено можливості застосування доповненої реальності та Unity у сфері розвитку особистості. Створено прототип нового продукту у сфері пошуку та організації заходів.

Практичне значення одержаних результатів

Отримана реалізація програмного продукту зможе допомогти користувачу у виборі варіантів проведення свого вільного часу та поширенні інформації про свої власні заходи.

Особистий внесок

1) Запропонований підхід до вирішення проблеми проведення вільного часу;
2) Створення прототипу програмного забезпечення, що змогло б вирішити проблему.

Публікації

За результатами наукових досліджень, проведених у бакалаврській роботі, підготовлено доповідь для участі в міжнародній науково-технічній фаховій конференції “8-ма Східно-Европейська конференція Математичні та програмні технології Internet of Everything”.

Структура та обсяг роботи

Робота викладена на 74 сторінках друкованого тексту, який складається із вступу, чотирьох розділів, висновків, списку використаних джерел. Робота містить 19 рисунків та 5 додатків.

РОЗДІЛ 1
АНАЛІЗ ПРОБЛЕМИ, ДОСЛІДЖЕННЯ В ПРЕДМЕТНІЙ ОБЛАСТІ

1.1Аналіз впливу відпочинку на здоров’я людини

Дослідження останніх років свідчать, що здоров'я кожного визначається умовами і способом життя більш ніж на 50%. З цього можна зробити висновок, що значна частина добробуту на якісного життя - це правильна організація дозвілля, у тому числі і рухової активності. Остання є безумовною складовою життєдіяльності. Саме організована рухова активність сприяє веденню здорового способу життя, попереджує розвиток ожиріння (від якого страждають 13% населення планети; 30% страждають від зайвої ваги). Також вона позитивно впливає на загальний рівень самопочуття: рух сприяє виробленню ендорфіну - так званого “гормону щастя”. Це все покращує імунітет, зменшує вплив шкідливих звичок на організм і позитивно впливає на психічний стан. Серед людей, що ведуть активний спосіб життя, значно менше спостерігаються асоціальні прояви та депресія.

Незважаючи на вищенаведені загальновідомі факти, найчастіше ми не дотримуємося норм фізичної активності: брак часу, офісна робота та інші чинники часто заважають проводити достатню кількість часу у русі. Активне впровадження комп'ютерних технологій різко знизило рухова активність як дітей, так і дорослих. Як наслідок, підвищився відсоток людей з ослабленим імунітетом.

1.1.1 Варіанти вибору типу відпочинку

Необхідний рівень рухової активності – обсяг і зміст рухової активності, що забезпечує природню потребу людини у русі задля зміцнення та забезпечення в подальшому свого здоров’я. Він сприяє не тільки загальному покращенню самопочуття, а ще й є профілактикою неінфекційних захворювань, підвищує працездатность. Саме тому у багатьох сучасних офісах зараз впровадили окремі кімнати-руханки чи спортзали: продуктивність прямо пов’язана з рухом. Але для якісного формування дозвілля, необхідно поєднувати різні види відпочинку.

1.1.1.1 Активний відпочинок

Спосіб проведення вільного часу, в процесі якого особа займається видами діяльності, що потребують активної фізичної роботи організму. Перехіід від одного виду діяльності до іншого(зміна активності) є елементарною потребою психофізичного здоров'я.

Приклади активного відпочинку: туризм, альпінізм, велоспорт, теніс, футбол, легка атлетика та інші.

1.1.1.2 Пасивний відпочинок

Відпочинок під час якого відсутня активна рухова діяльність та відбувається рекреація мозкових клітин. Це перегляд кінофільмів, сон, читання книг тощо. Основна мета такого відпочинку: надати спочуття відносного спокою.

1.1.1.3 Комбінований

Комбінований вид відпочинку - це синтез активного та пасивного відпочинку, у якому неможливо виокремити той чи інший стан. Під час такого відпочинку збільшується ефект відновлення працездатності.

Отже, можемо дійти висновку, що дозвілля за межами дому буде позитивно впливати на поєднання різноманітних форм відпочинку, посприяє розвитоку людини. Створений додаток дозволить зробити населення здоровішим, щасливішим та покращить загальний рівень життєдіяльності.

1.1.1.4 Інша класифікація структури вільного часу:

- індивідуальне(самостійне) ознайомлення з культурою. Сюди належить читання книг, журналів, газет, слухання радіо, перегляд телепередач тощо;
- публічно-видовищне споживання культури. Це відвідування театрів, кіно, концертів, музеїв, спортивних видовищ; такий вид відпочинку сприяє покращенню комунікаційних зв'язків між людьми;
- спілкування з членами сім'ї, родичами, сусідами, друзями та ін.;

- фізичні заняття (ранкова та вечірня гімнастика, заняття у спортивних секціях, йога, легка атлетика тощо);
- розваги та ігри для зняття розумового і фізичного напруження. Шляхом зміни діяльності, вони створюють гарний настрій, адже припиняється робота певних груп нервів чм м'язів, натомість, вводяться інші. - пасивний відпочинок (прогулянки, спокій, денний сон та ін.);
- заняття, які є явищами антикультури (зловживання алкоголем, хуліганство, наркоманія, паління);
- Прокрастинація (безцільне проведення часу, головна мета якого - відкласти важливу справу на потім). Наведена класифікація вільного часу охоплює лише найбільш поширені заняття, однак може включати будь-яку кількість видів діяльності. Найпоширеніші способи прокрастинації - безцільне використання соцмереж тощо.

1.2 Соціологічні дослідження проблеми

Найбільша проблема людини XXI століття - сидячий режим життя. Сучасність диктує необхідність роботи за комп'ютером, але потім люди часто обирають відпочинок за комп'ютером чи гаджетами. Як наслідок, спостерігаємо сумну статистику: тільки 35% молоді проводить свій вільний час подорожуючи, гуляючи на вулиці чи займаючись громадською активністю. Решта 65% проводять вільний час сидячи вдома, спілкуючись в Інтернеті чи дивлячись фільми. Дану інфографіку ми можемо побачити на рисунку 1.

Таким чином, ми можемо зробити висновок, що мільйони людей мають проблеми з організацією дозвілля, а саме: відсутність корисного відпочинку та культури організації вільного часу. Невдовзі це може призвести до багатьох хвороб,

які з’являться через такий ритм життя: найперше – ожиріння.

Рис. 1.1. Результати досліджень в Україні

Якщо ознайомитися із американською статистикою проведення вільного часу, то тенденція зберігається і люди більшість часу сидять вдома та займаються некорисними заняттями. Основна проблема полягає в нерівномірному комбінуванні різних видів відпочинку.

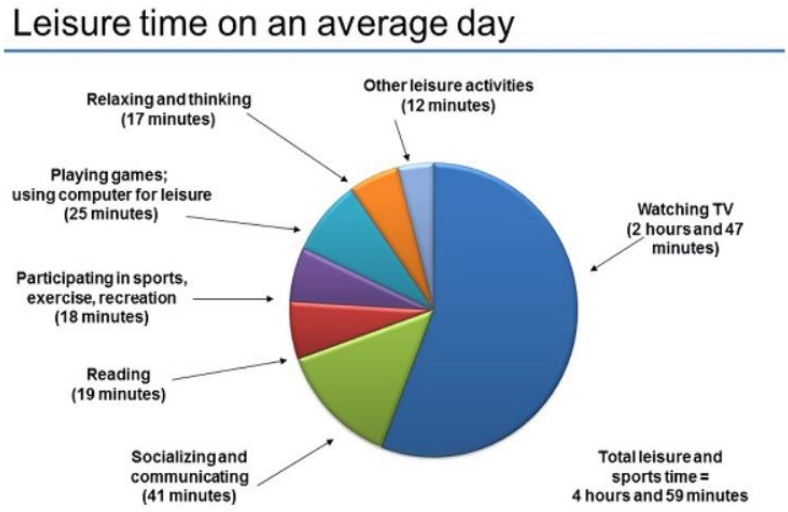


Рис. 1.2. Результати досліджень у США

1.3 Культура вільного часу

На попередній сторінці ми зустрілися з поняттям «культура вільного часу». Цей термін досить часто використовують соціологи, щоб звернути увагу на проблему наповнення(змісту) вільного часу, способу і міри його якісного освоєння з огляду на реалізацію і збагачення добробуту та якості життєдіяльності. Дане поняття важливе у контексті роботи тому що використовуючи додаток людина зможе розширити можливості проведення свого вільного часу, адже всі заходи та цікаві місця будуть показуватись в одному додатку.

Основу культури вільного часу становить свідома ніщо інше, як творча діяльність. Саме вона формує особистість зі здатністю розвиватися духовно і фізично. Культура вільного часу означає високий рівень його структури, яка визначається кількістю елементів і домінуючою роллю тих із них, які найбільше розвинені; певну міру поєднання рекреаційної і розвиваючої функцій вільного часу, активних і пасивних форм споживання духовних цінностей, спілкування і творчої діяльності; певну послідовність і тривалість видів діяльності, їх періодичність, частоту, ритм, насиченість, інтенсивність людської діяльності протягом певного відрізка часу; уміння економити час, запобігати нераціональним його затратам і досягати оптимального його обсягу.

Соціологи стверджують, що наявність «надлишкового» вільного часу, так само як і його «нестача», можуть негативно позначитись на формуванні корисного дозвілля.

1.4 Проблема просування та організації свого заходу

Незважаючи на безліч популярних соціальних мереж, проблема розповсюдженння інформації про створення немасштабних заходів є однією із найважливіших на етапі створення події, адже для цього потрібно залучити багато знайомих, та витратити немалу суму грошей.

Велика кількість різноманітних заходів часто проходить повз потенційно зацікавлених відвідувачів, які бажають підвищити культуру свого відпочинку. У чому найбільша проблема? Інформація про захід не “доходить” до потрібного кола потенційних відвідувачів.

Бачимо дві проблеми: 1) людина хоче організувати дозвілля;

2) людина-організатор хоче створити захід і запросити туди інших.

Таким чином, додаток зможе вирішити ці “болі” клієнтів. Він об’єднає людей, що зацікавлені у відвідуванні схожих заходів та допоможе створити їх самостійно. Це значно вплине на вирішення поставленої задачі, а також посприяє скороченню трудовитрат, пов’язаних з розвитком даної культурної сфери.

Проаналізувавши потреби користувачів та проблему організації власного заходу було виділено основні функціональні потреби програми :

1) Знаходження цікавих місць на мапі для проводження вільного часу за допомогою доповненої реальності або 3Д мапи (парк, клуб, кафе, події, що створені іншими користувачами);
2) Створити власну подію на мапі та поділитися з нею або з друзями, або зі всіма користувачами.
3) Створення статусу з бажанням про похід кудись;
4) Можливість додавати друзів та ділитися з ними планами та побажаннями;
5) Побудова маршруту до потрібного місця.

1.5 Висновки до розділу

Розібравши багато літератури, соціологічних опитувань, оцінивши ритми життя людей у XXI столітті було вирішено розробити прототип програмного забезпечення, який допоміг би людям комунікувати, активно проводити вільний час та доносити свої плани у зручний спосіб.

У розділі визначено основні функції програми та вимоги до неї. Програма являє собою мобільний додаток із зображенням мапи, де реалізовані та показані найближчі цікаві розважальні місця та події довкола.

РОЗДІЛ 2
ВИБІР СЕРЕДОВИЩА ПРОГРАМУВАННЯ ТА АНАЛІЗ МЕТОДІВ ДЛЯ СТВОРЕННЯ ПЗ

2.1 Аналіз технологій для створення додатку

2.1.1 Середовище програмування

Для програмування додатку було розглянуто дві різні методології для побудови програмного забезпечення :

1) створення продукту за допомогою нативного оточення. Це означало писати для платформи Андроїд зі збереженням швидкодії та детальної розробки на Java.

2) створення міні-гри за допомогою ігрового двигуна Unity та платформи .Net.

Були проаналізовані переваги та недоліки кожного з варіантів. Почнемо розгляд першого варіанту, а саме програмування додатку під Андроїд.

Переваги нативної розробки:

- Продуктивність. Особливо цей момент слід враховувати для мобільних версій. Така технологія економно використовує батарею, має збільшену швидкість роботи процесів. Все це можливо завдяки розробкам для окремих платформ. Тому цей метод дозволяє ефективно користуватися додатками.
- Інтерфейс. На відміну від першої «логіки» створення, код пишеться під конкретну систему, тому візуалізація чітко підлаштована під користувача, всі «кнопки» розташовуються на звичному місці.
- Доступ до службових програм. Тут немає обмежень на використання тієї ж геолокації або Bluetooth, що дає можливість вирішувати конкретні завдання.
- Тестування. Нативну програму або гру простіше тестувати. У цьому процесі використовуються всі механізми, які легко «зчитують» помилки і неточності. Вони ж, у наслідку, видаляються розробниками. Система оновлень на платформах

також включає в себе можливість утилізувати проблеми для конкретного користувача уже в процесі користування.

- Режим. Додаток, розроблений на нативній основі, працює як онлайн, так і без інтернету.
- Підтримка магазинів. Завантажуючи додаток з App Store або Google Play, користувач може бути впевнений в якості, так як при найменших невідповідності (графіка, звук, управління, візуальне наповнення) програма просто не пропускається в магазин, що позитивно відбивається на репутації самих магазинів і подібної технології. Більш того, завантаження програм з офіційного джерела є запорукою відсутності будь-яких перебоїв та вірусів.

Недоліки нативної розробки:

- Затратність грошей і часу. Над створенням одного додатку можуть працювати цілі команди. Це легко пояснити, адже для кожної версії, операційної системи потрібен фахівець, якому потрібно платити. Отже, збільшується бюджет, а також втрачається багато часу. Останній фактор слід також враховувати і заздалегідь підготуватися. Особливо це важливо розуміти і при оновленнях і підтримки - створюється не один додаток, а, як мінімум, два. Значить, і професіоналів також має бути більше.
- Відсутність сумісності. Такий стан є звичайним наслідком, так як для конкретної ОС пишеться код, який не може інтегруватися і працювати на іншій платформі. Наприклад, якщо програма створена для Android, то вона буде несумісна з IOS.
- Скорочення доходу. Це факт цілком реальний, адже гра або корисний календар пишеться тільки на одну платформу. Якщо в подальшому немає орієнтиру на інші системи, то частина ринку в будь-якому випадку втрачається, отже, зменшується і прибуток.

Вище було описані переваги та недоліки розробки нативно – орієнтованого продукту. Далі розберемо більш детальніше Unity.

Плюси використання Unity :

- Простота використання. Сюди варто віднести сама мова, яка є багаторівневою. Програмісту не потрібно наново щось вигадувати, адже мова вже реалізована. Від розробника потрібно тільки до одного об'єкту додавати інші. Наприклад, якщо створюється головний персонаж гри, то йому необхідні управління і зовнішня оболонка.
- Бюджетна розробка. Бюджет стає значно меншим, завдяки тій же платформі. Код працює відразу на всіх пристроях, отже, зменшується кількість самих розробників і робочих годин. Відповідно зменшується і платня. Сам двигун є безкоштовним. Звичайно, є платна підписка, але для більшості вона може і не знадобитися.
- Час. Виходячи з того ж «перетину», час на створення об'єкта зменшується. Зменшенню годин сприяє і відсутність унікальних елементів інтерфейсу і єдина технологія.
- Підтримка. Дана технологія має одне із найсильніших ком’юніті. На офіційному сайті знаходиться вся інформація по кожному кроку розробки. Якщо виникло питання і, здається, що на нього немає відповіді, то варто зайти на сайт і відразу ж зрозуміти, в чому справа. Навіть якщо відсутній конкретний приклад, то ніхто не відміняв службу підтримки.
- Asset Store. У цьому місці знаходяться всі необхідні ресурси для створення програми. Зручний пошук і сортування дозволяють швидко підібрати те, що потрібно.
- Єдина логіка. Одна технологія дозволяє, по-перше, проводити оновлення одночасно на всіх пристроях, по-друге, уникнути подвійної роботи, так як єдина система містить меншу кількість помилок.

Недоліки Unity 3D

Ідеальним бути нічого не може, тому й тут не обійшлося без мінусів. Повільність. Цей фактор часто випливає з таких додатків, які мають на увазі масштабність, складні сцени. Доводиться використовувати додаткові утиліти, і все це впливає на продуктивність.

- «Ваговитість». Практично всі програми мають великий обсяг. Якщо мова йде про мобільний додаток, то це може зайняти багато пам'яті в телефоні. Навіть якщо в просту гру включити мінімум, тобто звук, графіку, то інсталяційний файл буде важити близько 20 МБ.
- Відсутність «самостійності». Воно полягає в тому, що при використанні Unity автоматично вбудовується їх логотип. Позбутися його можна сплативши PRO-версію.
- Обмеження. Воно являє собою відсутність функцій управління «вшитими» параметрами, починаючи від камери і закінчуючи базою даних. Це робить додаток «негнучким» і виключає роботу на максимальних можливостях.

Таким чином, проаналізувавши два способи створення програмного забезпечення було виділено головні переваги кожного способу, порівняно їх та обрано ігровий двигун Unity. Основним критерієм слугувала кросплатформність отриманого продукту, адже це значно впливатиме на час розробки.

Головним середовищем розробки буде Unity.

Середовище розробки Unity дозволяє використати різні додаткові бібліотеки, готові збірки для покращення свого продукту. Відкриває перед користувачем безмежний світ створюваної графіки, яку можна додати до програмного забезпечення та зробити його максимально наближеним до гри із різними ефектами та функціоналом.

2.1.2 Доповнена реальність

Можливі варіанти для реалізації доповненої реальності у ПЗ :

- Tango - це бренд, а не продукт. Він складається з апаратного референсного дизайну (RGB і фишай-лінзи, датчик глибини і деякі специфікації CPU / GPU) і програмного стека, який забезпечує VIO (відстеження руху), Sparse Mapping (вивчення області) і щільну 3D-реконструкцію (сприйняття глибини).
- Hololens має той же самий стек програм, але включає і деякі ASIC (які вони називають блоками голографічного обробки, Holographic Processing Units), щоб розвантажити CPU і GPU.
- ARKit - це система розвитку доповненої реальності високого рівня, яка використовує обчислювальну потужність на високоефективних, але потужних пристроях iOS та їх камерах
- Vuforia пропонує приблизно те ж саме, але має своє обладнання.

Всі вони використовують одну і ту ж систему VIO (Tango і ARKit навіть одну і ту ж базу коду, розроблену FlyBy). Ні Hololens, ні Tango не використовують камеру глибини для відстеження, хоча в майбутньому це покращило б роботу програм, побудованих на даних системах.

Проаналізувавши можливі пакети з доповненої реальності під Unity було обрано ARCore. Це інструментарій Google для створення додатків доповненої реальності. Найкраще в ARCore те, що він підтримує розробку як для Android (7.0 і вище), так і для платформ iOS (11 або вище). ARCore пропонує точки, виявлення площини, позу, оцінку освітленості, якоря, відстеження зображень, відстеження осіб, оклюзію об'єктів і хмарні якоря.

Рис. 2.1. Технології проекту

Базові поняття роботи ARCore :

1) Відслідковування руху. Коли телефон переміщується, ARCore використовує процес, званий одночасної локалізацією і відображенням, або SLAM, щоб зрозуміти, де ви перебуваєте щодо світу навколо нього. ARCore виявляє візуально відмінні особливості на знімку з камери, звані характерними точками, і використовує ці точки для обчислення зміни свого місця розташування. Візуальна інформація в поєднанні з інерційних вимірювань від пристрою для оцінки пози (положення і орієнтації) камери щодо світу протягом довгого часу.

2) Розуміння навколишнього середовища. ARCore постійно покращує своє розуміння навколишнього середовища реального світу, виявляючи характерні точки і площини. ARCore шукає кластери характерних точок, які здаються лежать на загальних горизонтальних або вертикальних поверхнях, таких як столи або стіни, і робить ці поверхні доступними для вашого застосування в вигляді площин. ARCore також може визначати межу кожної площині і надавати цю інформацію додатку.

Можна використовувати цю інформацію для розміщення віртуальних об'єктів на плоских поверхнях.

3) Розуміння глибини. ARCore може створювати карти глибини, зображення, які містять дані про відстані між поверхнями від заданої точки, використовуючи основну камеру RGB з підтримуваного пристрою. Можна використовувати інформацію, надану картою глибини, щоб забезпечити імерсивні і реалістичний призначений для користувача досвід, наприклад, змусити віртуальні об'єкти точно стикатися з спостерігаються поверхнями або змусити їх з'являтися перед або за об'єктами реального світу.

4) Оцінка світла. ARCore може виявляти інформацію про висвітлення навколишнього середовища і надавати вам середню інтенсивність і колірну корекцію для даного зображення камери. Ця інформація дозволяє висвітлювати віртуальні об'єкти в тих же умовах, що і їх навколишнє середовище, підвищуючи відчуття реалізму.

5) Взаємодія з користувачем. ARCore використовує перевірку попадання, щоб взяти координату (x, y), що відповідає екрану телефону (надану киснем або будь-яким іншим взаємодією, яке ви хочете, щоб ваше додаток підтримувало), і проектує промінь в поле зору камери на світ, повертаючи будь-які літаки або характерні точки, які перетинає промінь, разом з позою перетину в світовому просторі. Це дозволяє вибирати об'єкти в середовищі або іншим чином взаємодіяти з ними.

6) Точки орієнтування. Дозволяють розміщувати віртуальні об'єкти на похилих поверхнях. Коли ви виконуєте хіт-тест, який повертає характерну точку, ARCore буде дивитися на прилеглі характерні точки і використовувати їх, щоб спробувати оцінити кут поверхні в даній характерній точці. Потім ARCore поверне позу, яка враховує цей кут. Оскільки ARCore використовує кластери характерних точок для визначення кута поверхні, поверхні без текстури, такі як біла стіна, можуть не виявлятися належним чином.

Технічно ARCore - система візуальної інерційної одометрії (VIO) з простим визначенням 2D площини. VIO позначає, що програмне забезпечення відстежує позицію користувача в просторі (вашу 6dof-позицію) в реальному часі, тобто обчислює її з кожним оновленням дисплея, 30 або більше разів на секунду. Ці обчислення проводяться двічі, паралельно. Ваша позиція відстежується через візуальну систему (камеру), порівнюючи точку в реальному світі з пікселями на сенсорі камери. Також вона відстежується інерційної системою (вашим акселерометром і гіроскопом - інерційної вимірювальної одиницею або IMU). Вихідні дані цих двох систем поєднуються через фільтр Калмана, який визначає, яка з двох систем надає кращу оцінку вашої справжньої позиції (ground truth) і публікує це оновлення позиції через SDK ARCore. Як одометр в автомобілі визначає відстань, яку проїхав автомобіль, система VIO відстежує відстань, яке телефон користувача пройшов в 6D-просторі. 6D об'єднує 3D вимірювання по осях x, y і z (переміщення), а також крен, рискання і обертання.

Великою перевагою, яке пропонує VIO, є те, що свідчення IMU виробляються приблизно 1000 разів в секунду і засновані на прискоренні. Для вимірювання руху пристрою між показаннями IMU використовується навігаційне числення. Помилки в інерціальній системі накопичуються з плином часу, тому чим більше часу проходить між кадрами IMU або чим довше інерційна система працює без отримання даних візуальної системи, тим більше відстеження буде відхилятися від земельної поверхні.

Візуальні / оптичні вимірювання проводяться з частотою кадрів камери, як правило це 30 кадрів в секунду, і засновані на відстані (зміни сцени між кадрами). Оптичні системи зазвичай накопичують помилки в відстані (і часу в меншій мірі), тому чим далі ви переміщується, тим більше помилка. Але незважаючи на це, сильні сторони кожної системи скасовують недоліки іншої.

Тож візуальна і інерційна системи засновані на абсолютно різних системах виміру, незалежних один від одного. Це означає, що камера може бути закрита або показувати сцену на кшталт білої стіни, і інерційна система буде підтримувати

картинку ще кілька кадрів. Або пристрій може бути в нерухомому стані, а візуальна система визначить позицію більш стабільно, ніж інерційна система. Фільтр Калмана постійно вибирає більш точну позицію і призводить до стабільного відстеження.

Що найцікавіше, системи VIO існують вже багато років, їх добре розуміють в індустрії, і на ринку існує кілька застосувань для них. Тому той факт, що Google використовує VIO, значить небагато. Ми повинні подивитися, чому ця дана система така надійна.

Друга важлива частина ARCore - це просте визначення площині. Це необхідно для створення поверхні, на якій ви розміщуєте зміст, інакше все буде просто парити в просторі. Вона обчислюється на основі властивостей, певних оптичною системою (маленькі точки, які ми можемо побачити в демо версії), і алгоритм просто усереднює їх таким чином, що три точки визначають поверхню, і якщо ви робите це достатня кількість разів, ви можете оцінити місце розташування справжньою поверхні . Ці точки називаються "хмара точок", і для оптичного відстеження використовуються зріджені хмари точок. Вони використовують набагато менше пам'яті і процесорного час для відстеження, а за підтримки інерціальній системі оптична система може ефективно функціонувати з невеликою кількістю точок для відстеження. Інший тип - це щільна хмара точок, яке виглядає ближче до реальності.

2.2 Забезпечення коректної роботи програми з ARCore

Для коректної роботи додатку важливо мати коректно налаштовану систему візуальної інерційної одометрії (VIO). Почнемо з оптичного калібрування та інерційного.

2.2.1. Оптичне калібрування камери:

Оптичне калібрування: воно використовує модель камери-обскури для корекції поля зору лінзи і таких речей, як «ефект бочки» лінзи. Майже кожне зображення деформується через форми лінзи. Багато розробників можуть впоратися з цим кроком без співпраці з виробниками, використовуючи базові публічні специфікації камери.

Фотометричне калібрування: воно вимагає участі виробника, так як стосується особливостей самого сенсора зображень, покриттів зовнішніх лінз і т.д. Ця калібрування має справу з мепінгом кольору та інтенсивностю. Наприклад, камери телескопа, фотографують далекі зірки, повинні знати, чи є зміна в інтенсивності світла на пікселі зіркою або це просто відхилення в сенсорі лінзи. Це калібрування призводить до набагато більшої впевненості в тому, що піксель на сенсорі співвідноситься з точкою в реальному світі, і, таким чином, оптичне відстеження стає набагато надійніше.

2.2.2. Інерційне калібрування

Важливо пам'ятати, що IMU вимірює прискорення, а не відстань або швидкість, а помилки у вимірах IMU швидко накопичуються. Мета калібрування і моделювання - переконатися, що вимір відстані досить точно для X доль секунди. В ідеалі цей період покриває час, який камера припиняє відстеження на пару кадрів, або користувач закриває камеру, або щось ще відбувається в кадрі.

Вимірювання відстані за допомогою IMU називається навігаційним численням. По суті, це здогад, але цей здогад уточнюють за допомогою моделювання поведінки IMU і створення фільтрів для усунення помилок. Уявіть, якби вас попросили зробити крок, а потім вгадати його довжину в дюймах. Один крок і здогад приведуть до високого ступеня помилки. Якщо ви послідовно зробите тисячі кроків, виміряєте кожен і навчитеся враховувати те, з якої ноги ви зробили крок, покриття підлоги, взуття, швидкість руху та інші параметри, ваші здогадки поступово стануть дуже точними. Це і відбувається при калібрування IMU і моделюванні.

Існує багато джерел помилок. Роботичну руку використовують для переміщення пристрою одним і тим же чином знову і знову до тих пір, поки дані IMU не збігатимуться з істинним значенням від роботичної руки. Google і Microsoft навіть відправляли свої пристрої в умови мікрогравітації на МКС, щоб усунути додаткові помилки.

Отримати дійсно точну систему навіть складніше, ніж може здатися. Виробники повинні проходити цей процес з усіма пристроями в портфоліо, але навіть тоді у різних пристроїв можуть бути різні IMU. Наприклад, в Galaxy 7 може бути IMU від Invensense або від Bosch, і моделювання двох пристроїв працює по-різному.

2.3 Використання мапи в додатку

Для розробки додатку потрібно було обов'язково використовувати мапу, для кращої навігації користувачів. Тому було проведено дослідження декількох продуктів на ринку та порівняно їх функціонал, ціну та зручність використання у даному проекті.

Такими конкурентами були декілька видів мап, а саме Google Maps, Mapbox, HERE.

Першим критерієм для вибору була деталізація зображення вулиць на мапі. Були проведені тестові експеременти, де вибиралась одна місцевість та порівнювалась на різних картах. Для досліду було обрано село Крушинка Васильківського району Київської області. Різницю якості зображень зображено (Див. Додаток Г).

Результат : Mapbox та HERE мають найкращий рівень деталізації місцевості, адже Google Maps не зміг намалювати навіть назви вулиць даного села.

Інший критерій – це можливість кастомізувати карти та модернізувати їх для своїх цілей. Таким чином Mapbox та Google Maps мають найбільший функціонал у розширенні варіацій карти.

Критерій № 3 – ціна використання продукту. У даному випадку Google Maps є найдорожчим, адже вимагає 7$ за кожні 28 000 завантажень карти користувачами. Mapbox - 5$ за кожні 100 000 завантажень. Ціноутворення HERE не повністю зрозуміло, адже ціна залежить від пакетів транзакцій. Стандартний коштує 0,5$ за 1000 підключень.

Так можна виділити основні переваги Mapbox :

- Унікальні параметри налаштування. Mapbox налаштовується більше, ніж Карти Google. Безліч систем картографування пропонують готову карту, Mapbox - це як коробка з лего. Mapbox дозволяє створити стиль карти, який відповідає темі вашого додатку. Розробник може встановити шрифти та колірну схему та додати функціональні можливості, такі як покрокові вказівки та інформація про місцевість.
- Це проект з відкритим програмним кодом. Mapbox ділиться своїм кодом на GitHub, щоб його завжди можна було побачити, проаналізувати та покращити. Багато талановитих розробників активно вносять свій внесок у кодову базу. SDK-карти карт Mapbox засновані на Mapbox GL Native. Ця бібліотека дозволяє вбудовувати інтерактивні, настроювані векторні карти в рідні програми на кількох платформах.
- SDK для карт Mapbox SDK для Unity дозволяє будувати враження на основі розташування, використовуючи визначні місця (POI) у всьому світі. Розробник можете додати місцеположення за допомогою перетягування карт та POI, 3D-будівель та місцевості за допомогою технологій доповненої реальності.
- Ще однією перевагою даної технології є офлайн карти. Даного режиму немає у API Google карт. Точніше, офлайн режим доступний у фірмовій програмі Google Maps; однак він обмежений самим додатком і недоступний через API, тому його не можна інтегрувати в інші продукти. Mapbox забезпечує більшу гнучкість щодо режиму офлайн. Завдяки використанню векторних карт Mapbox підтримує функцію офлайн. Програми, створені за допомогою мобільних SDK Mapbox, можуть завантажувати карти для вибраних географічних районів для використання, коли пристрій не має підключення до мережі. Крім того, мобільні SDK Mapbox автоматично кешують райони на карті та інші ресурси, запитувані під час звичайного використання.

За допомогою яких технологій працює Mapbox?

Дані беруться з відкритих джерел, таких як OpenStreetMap і NASA, а також з придбаних власних джерел даних, таких як DigitalGlobe. Технологія заснована на Node.js, Mapnik, GDAL і Leaflet. Mapbox використовує анонімізовані дані з телеметричних пінгів, таких як Strava і RunKeeper, для ідентифікації ймовірних відсутніх даних в OpenStreetMap за допомогою автоматичних методів, потім вручну застосовує виправлення або повідомляє про проблему розробникам OSM.

Таким чином було обрано для розробки мапу Mapbox, яка є зручною у користуванні, легко піддається змінам, не є дорогою в обслуговуванні та має великий функціонал та популярність серед розробників.

2.4 Висновки до розділу

У розділі було проведено аналіз технологій, які могли б бути використані у програмуванні додатку. Основним кістяком програми буде Unity та мова програмування C#. Додатково для реалізації відображення карти буде задіяно API Mapbox. Складність вибору інструменту реалізації доповненої реальності полягала у схожості конкурентних пропозицій та різноманітності пропозицій на ринкую. Проаналізувавши всі варіанти вирішено застосувати ARCore. Було оцінено власну можливості та проаналізувавши всі плюси та мінуси кожної технології було обрано оптимальну зв’язку для побудови програми.

РОЗДІЛ 3

СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Для вирішення проблеми було створено програмне забезпечення за допомогою представлених вище технологій, а саме Unity, Mapbox, ARCore та .Net платформи.

3.1. Створення методів огляду навколишнього середовища

Почнемо з того, що було створено дві камери – спостерігачі у даному додатку. Перша камера для доповненої реальності, що передає зображення у даному форматі :

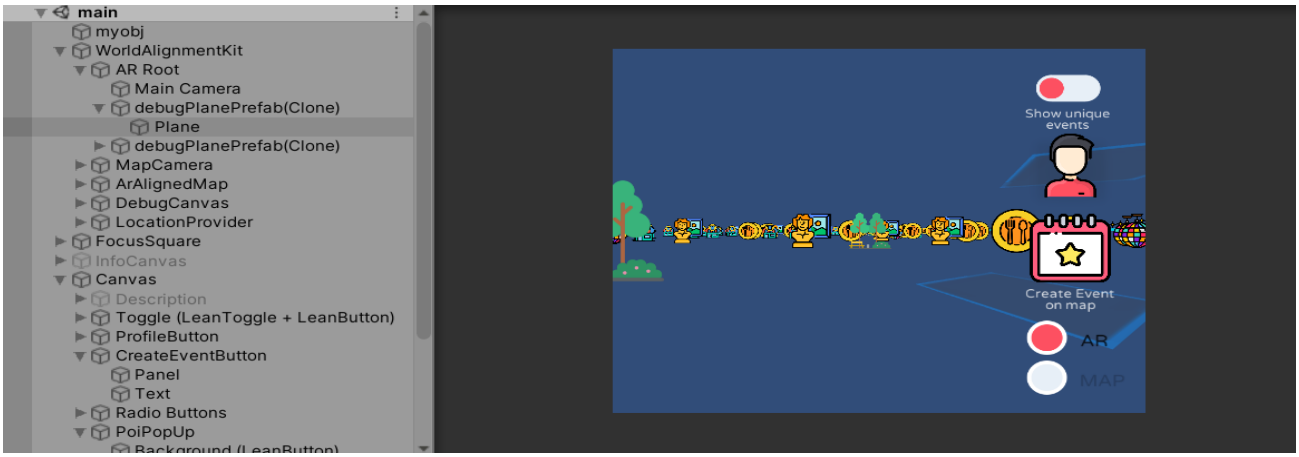


Рис. 3.1. Розробка у середовищі Unity режиму AR

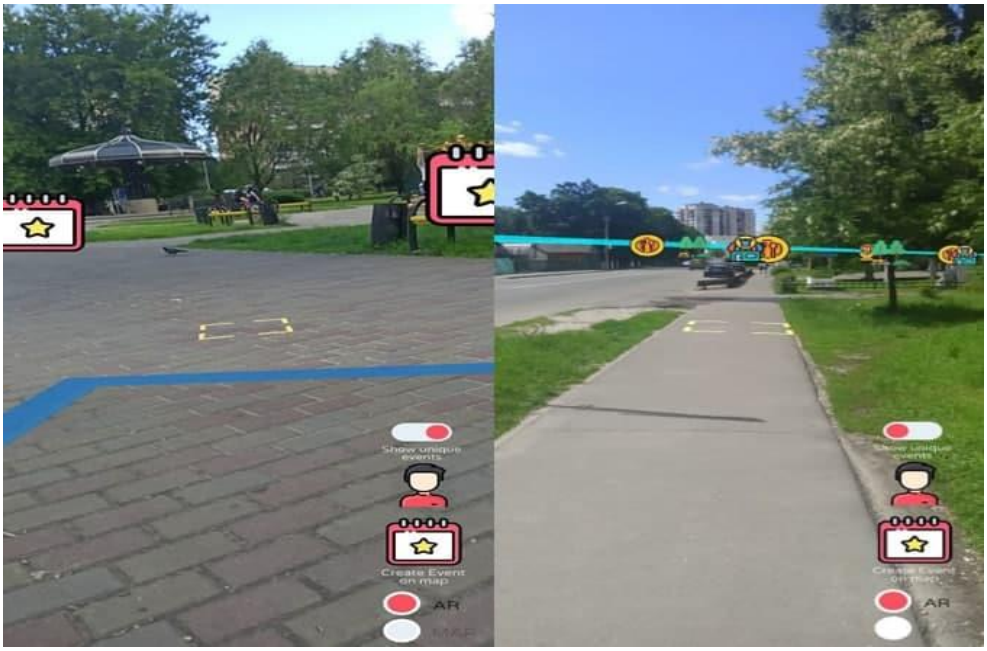


Рис. 3.2. Вигляд у житті

Камера спостерігача від третьої особи, що передає зображення як 3Д карту, яка виглядає так :

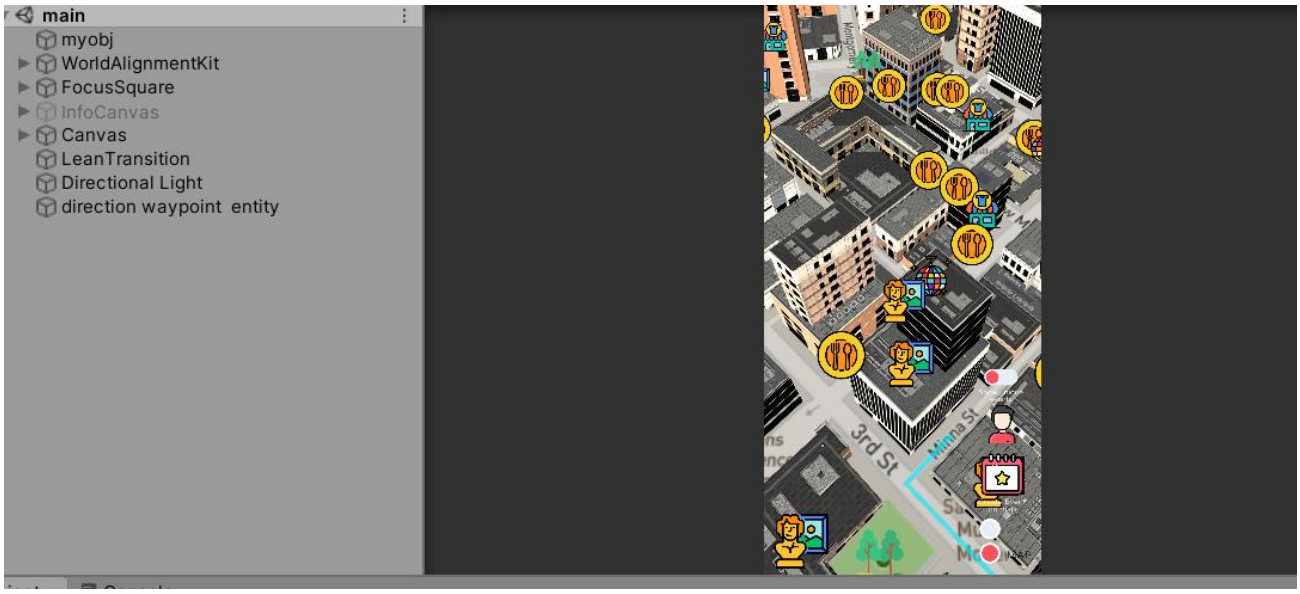


Рис. 3.3. Розробка у середовищі Unity режиму 3Д карти

Рис. 3.4. Вигляд 3Д карти на смартфоні

Для зручнішої фільтрації подій на мапі було створено перемикач, що залишає тільки події, створені іншими користувачами див. рис. 3.5.

Рис. 3.5. Фільтр подій

3.2 Створення карти та її налаштування

Для візуалізації принципу побудови та роботи мапи представимо карту на діаграмі компонентів.

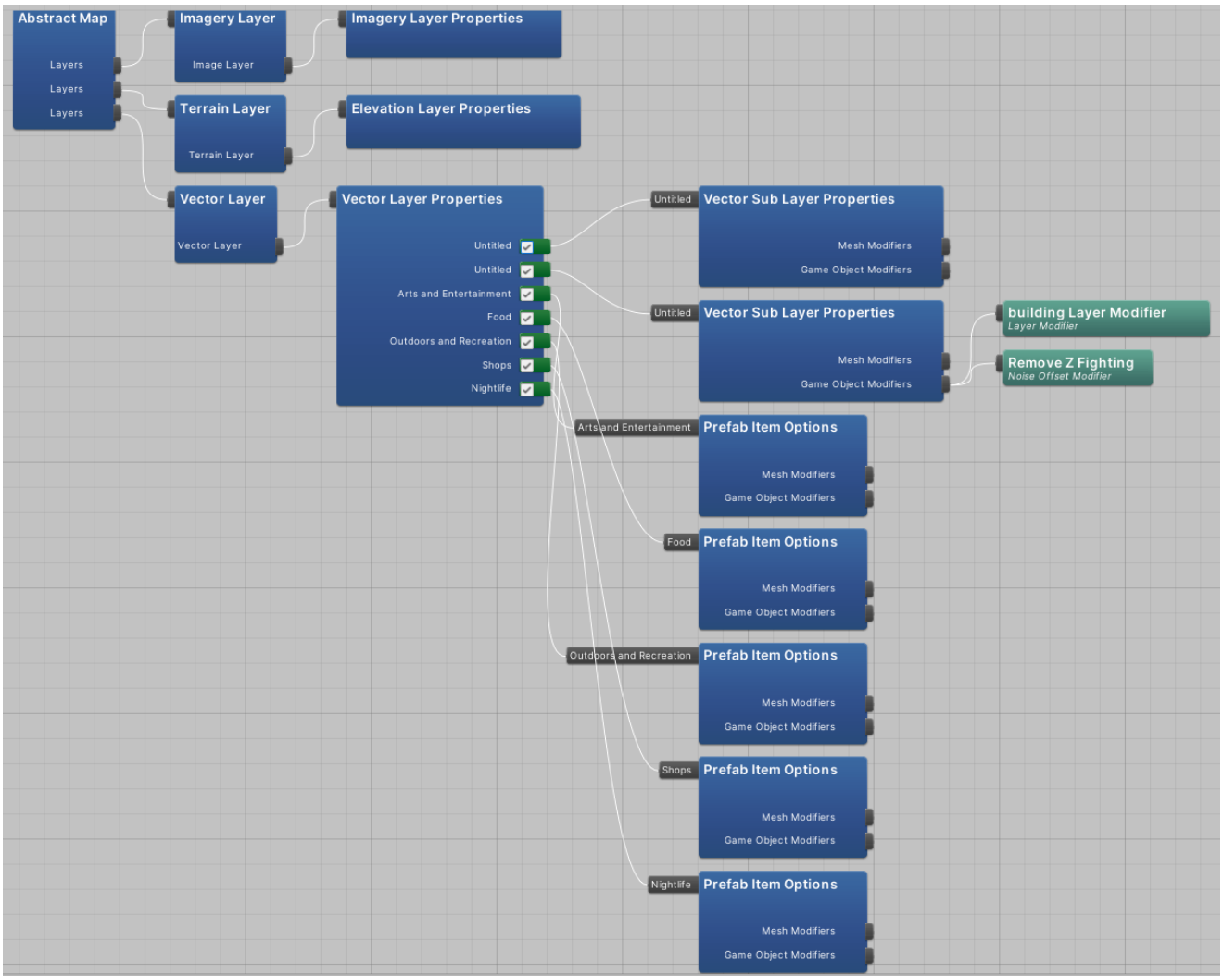


Рис. 3.5 – Діаграма компонентів карти

Головний опис карти створюється в скрипті Abstract map, де задаються початкові координати камери, персоніфікатор карти, обирається вид ландшафту, задаються слої та точки інтересу на мапі. В даному скрипті описується візуальна частина. Точки інтересу розбиваються на об’єкти – префаби. Це унікальний тип ассетів, що дозволяє зберігати весь об’єкт із всіма компонентами та значеннями властивостей. Тобто він дозволяє редагувати всі об’єкти відразу. У даному випадку ми маємо багато різновидів розваг і це дозволяє підібрати під кожен вид свою іконку, і описати їх по різному. Тобто будь-яка зміна у префабі відразу відображається на всіх його екземплярах. У програмі задіяно 6 видів префабів, щоб описати кожен із різновидів розваг.

Код, що описує кожен із префабів (Див. Додаток А).

Інформація щодо різновидів іконок – подій (Див. Додаток В).

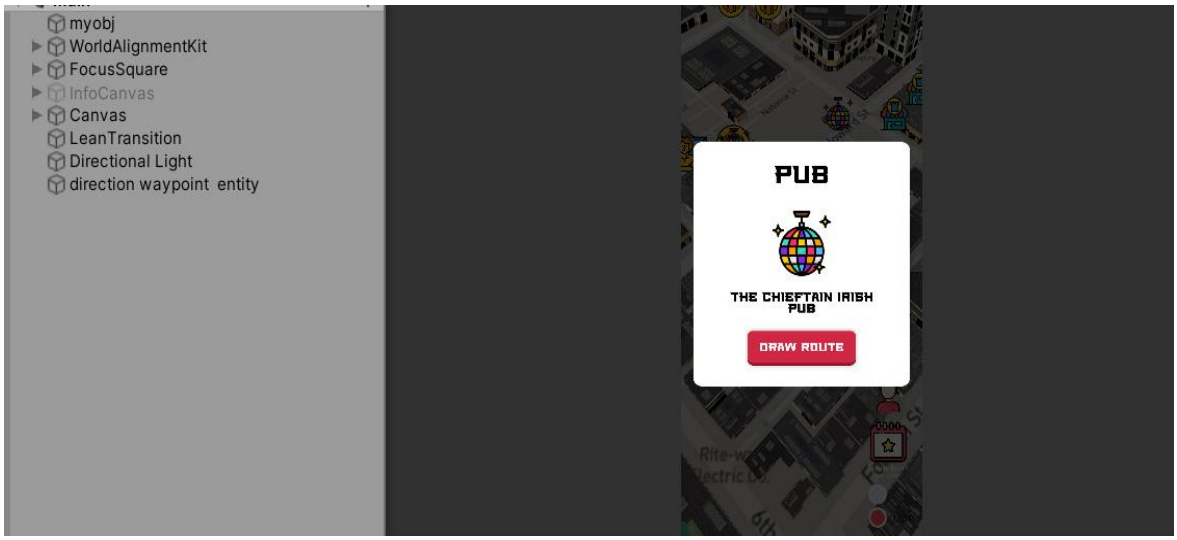


Рис. 3.6. Розробка іконок

Під час роботи програми натискання на будь – яку точку інтересу викликає додаткове вікно, що виводить інформацію про дану точку. У житті це виглядає так :

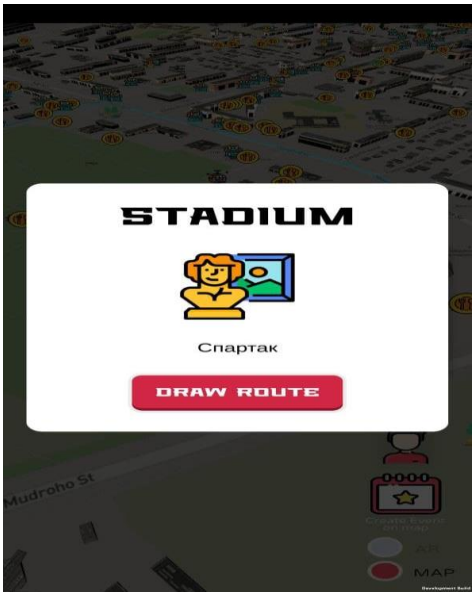


Рис. 3.7. Запуск на смартфоні

У додатку ефект натискання побудований за технологією Raycaster. У даному випадку в точці натискання випускається промінь горизонтально поверхні і якщо на його шляху зустрінеться картинка з написом про подію, то відкривається додаткова інформація про нього. Вона реалізується за допомогою коду (Див. Додаток Б).

3.3 Побудова маршруту

Однією із основних по зручності функцій є побудова маршруту для користувача. Адже обрати якесь місце легко, а знайти найкоротший маршрут до нього - ні. Тобто потрібно додати обов’язково даний функціонал до програми.

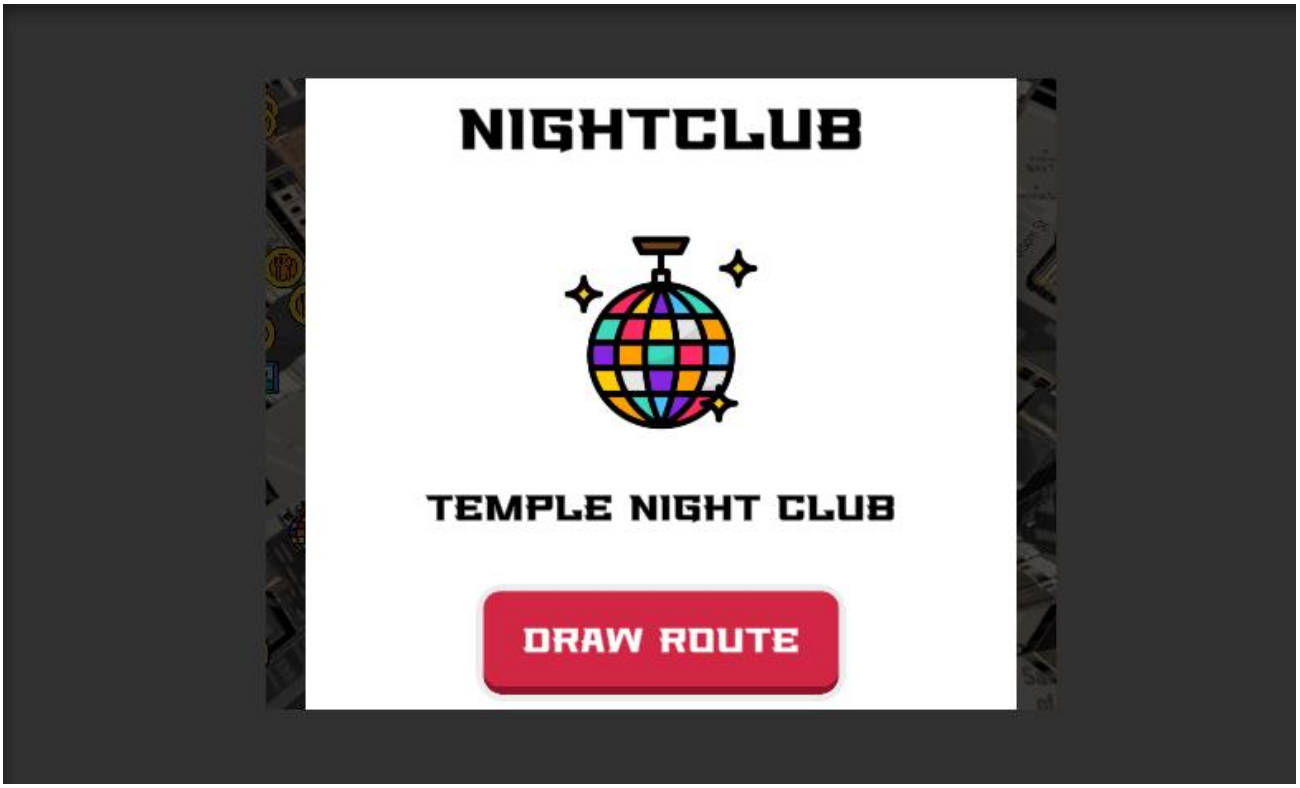


Рис. 3.8. Зображення інтерфейсу побудови маршрута

Рис. 3.9. Побудова маршруту до заданої події

3.3 Порядок виконання функцій та подій у програмі

3.3.1 Перша загрузка сцени

Дані функції викликаються одразу при запуску сцени (один раз для кожного об’єкта на сцені).

Awake: Ця функція завжди викликається до будь-яких функцій Start і також після того, як префаб був викликаний в сцену (якщо GameObject неактивний на момент старту, Awake НЕ буде викликаний, поки GameObject не буде діяти, або функція в якомусь прикріпленому скрипті не викличе Awake).

OnEnable: (викликається тільки якщо об'єкт активний): Ця функція викликається відразу після включення об'єкта. Це відбувається при створенні зразка MonoBehaviour, наприклад, при завантаженні рівня або був викликаний GameObject з компонентом скрипта.

OnLevelWasLoaded: Ця функція виконується щоб інформувати, що новий ігровий рівень загрузився.

Варто врахувати, що для об'єктів, доданих в сцену відразу, функції Awake і OnEnable для всіх скриптів будуть викликані до виклику Start, Update і т.д. Природно, для об'єктів викликаних під час ігрового процесу такого не буде.

3.3.2 Перед першим оновленням кадру

Start: Функція Start викликається до поновлення першого кадру тільки якщо скрипт включений.

Для об'єктів доданих на сцену, функція Start буде викликатися у всіх скриптах до функції Update. Природно, це не може бути забезпечено при створенні об'єкта безпосередньо під час гри.

Між кадрами

OnApplicationPause: Ця функція викликається в кінці кадру, під час під час якого викликається пауза, що ефективно між звичайними оновленнями кадрів. Один додатковий кадр буде виданий після виклику OnApplicationPause, щоб дозволити грі відобразити графіку, яка вказує на стан паузи.

3.3.3 Порядок поновлення

Коли ви відстежуєте ігрову логіку і взаємодії, анімації, позиції камери є кілька різних подій, які ви можете використовувати. За загальним шаблоном, велика частина завдань виконується всередині функції Update, але є також ще інші функції, які ви можете використовувати.

FixedUpdate: Найчастіше трапляється, що FixedUpdate викликається частіше ніж Update. FU може бути викликаний кілька разів за кадр, якщо FPS низький і функція може бути і зовсім не викликана між кадрами, якщо FPS високий. Всі фізичні обчислення і оновлення відбуваються відразу після FixedUpdate.

Update: Update викликається раз за кадр. Це головна функція для оновлень кадрів.

LateUpdate: LateUpdate викликається раз в кадр, після завершення Update. Будь-які обчислення зроблені в Update будуть вже виконані на момент початку LateUpdate.

Часто LateUpdate використовують для наступної камери від третьої особи. Якщо ви переміщаєте і повертаєте персонажа в Update, ви можете виконати всі обчислення переміщення і обертання камери в LateUpdate. Це забезпечить те, що персонаж буде рухатися до того, як камера відстежить його позицію.

3.3.4 Рендеринг

OnPreCull: Викликається до того, як камера відсіче сцену. Відсікання визначає, які об'єкти будуть видні в камері. OnPreCull викликається прямо перед тим, як починається відсікання.

OnBecameVisible / OnBecameInvisible: Викликається тоді, коли об'єкт стає видимим / невидимим будь-якій камері.

OnWillRenderObject: Викликається один раз для кожної камери, якщо об'єкт в поле зору.

OnPreRender: Викликається перед тим, як камера почне рендерити сцену.

OnRenderObject: Викликається, після того, як всі звичайні рендери сцени завершаться. Ви можете використовувати клас GL або Graphics.DrawMeshNow, щоб малювати призначену для користувача геометрію в даній точці.

OnPostRender: Викликається після того, як камера завершить рендер сцени.

OnRenderImage (тільки в Pro версії): Викликається після завершення рендеру сцени, для можливості пост-обробки зображення екрану.

OnGUI: Викликається кілька разів за кадр і відповідає за елементи інтерфейсу (GUI). Спочатку обробляються події макета і розмальовки, після чого йдуть події клавіатури / мишки для кожної події.

OnDrawGizmos Використовується для відтворення Гизмо у вікні Scene View з метою візуалізації.

3.3.5 При виході

Ці функції викликаються в усіх активних об'єктах у сцені:

OnApplicationQuit: Ця функція викликається для всіх ігрових об'єктів перед тим, як додаток закривається. У редакторі викликається тоді, коли гравець зупиняє ігровий режим. У веб-плеєрі викликається по закриття веб вікна.

OnDisable: Ця функція викликається, коли об'єкт відключається або стає неактивним.

Загальну картину виконання програми можна побачити на рисунку 3.8.

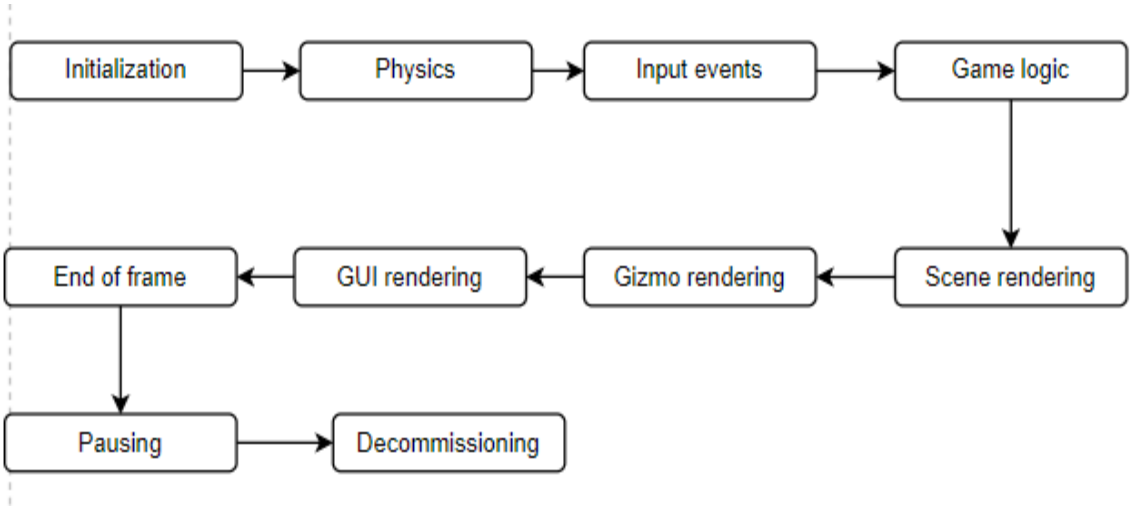


Рис 3.8. Порядок виконання програми

3.4 Слої

Даний метод у створеному програмному забезпеченні використовується для фільтрації зображень на мапі. Таким чином ми зручно можемо видалити будь-яку інформацію з мапи видаливши тільки шар. Очевидно, що для зручності даної функціональності потрібно розумно спроектувати шари за їх властивостями, щоб не видаляти зайвого.

Використані слої можна побачити на рисунку 3.9.

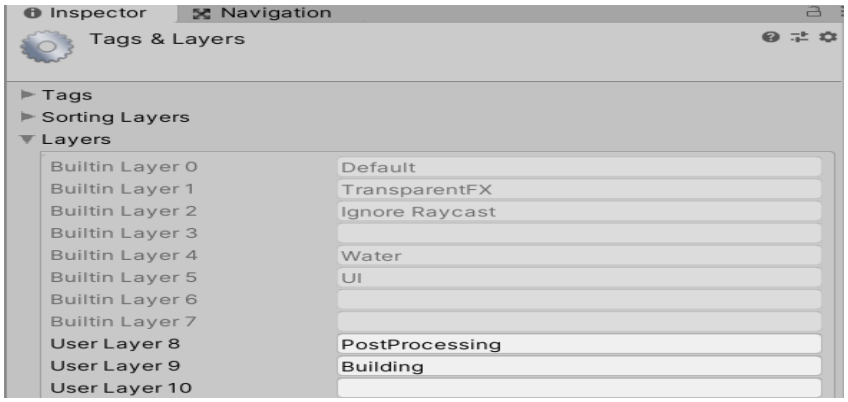


Рис. 3.9. Використані слої

В програмі активно застосовується створення сцени за допомогою culling mask.

При переході зі звичайної камери на камеру відображення доповненої реальності зникає слой будівель і залишаються лише позначки з інформацією про цікаві об’єкти.

Дане перемикання між камерами відбувається за допомогою кнопки, що зображена в правому нижньому куті екрану.

Рис. 3.10. Кнопки перемикання режиму

3.5 Висновки до розділу

У даному розділі було показано та пояснено основні моменти створення програмного забезпечення. Створено алгоритм взаємодії всіх технологій проекту, пов’язано роботу віртуальної реальності зі звичайними картами. Розроблено функцію побудови маршруту до потрібної точки. Більше технічної інформації знаходиться в SAD (Додаток Д).

Результатом виконання розділу є створення прототипу програмного продукту.

РОЗДІЛ 4
ІНТЕРФЕЙС КОРИСТУВАЧА

4.1 Опис основних частин інтерфейсу

Програма має простий та зрозумілий інтерфейс користувача, котрий складається із декількох кнопок. Це кнопка Profile, тобто профіль користувача, який містить коротку інформацію про нього та кнопка Create Event on map, яка відповідає за створення власної події на мапі. Дана кнопка працює у режимі 3Д мапи та при натисканні дає можливість користувачу обрати місце на карті та розташувати позначку із запланованою подією.

Детальніше про функціонал кнопки Profile. При натисканні на іконку Profile користувач отримує вікно :

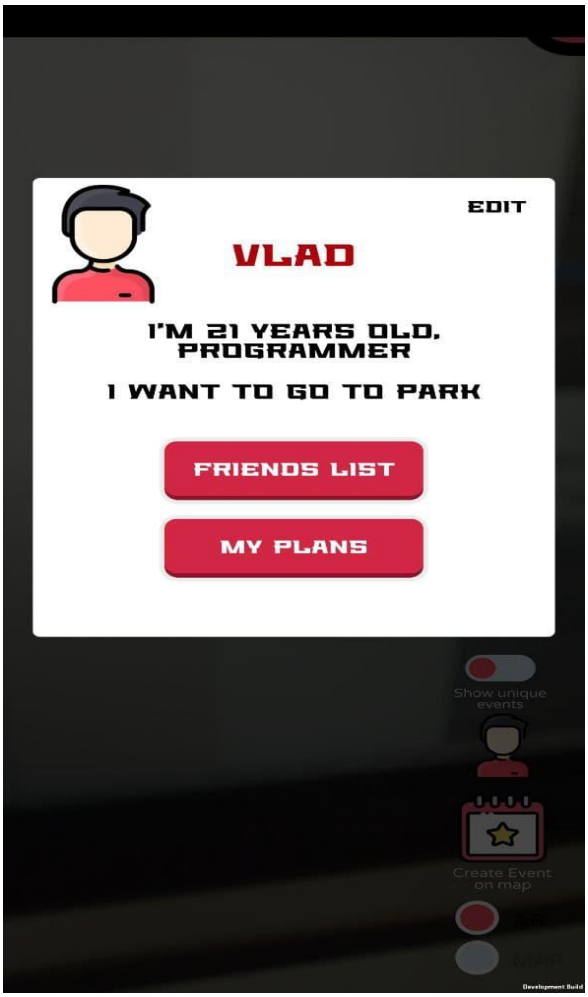


Рис. 4.1. Інформація про профіль

У даному вікні користувач зможе ввести своє ім’я у полі Name, коротку інформацію у полі About та настрій на тип розваг у полі Status.

Також при натисканні на кнопку Profile користувач може зайти до списку своїх друзів та подій.

Детальніше про кнопку Create Event on map :

Рис. 4.2. Кнопка для створення подій

Дана кнопка дозволяє користувачу створити свій особистий значок на карті та запланувати якусь подію. Як використовувати ? У режимі мапи натиснути на значок події, а потім на бажану точку на карті. Відразу з’явиться мітка, яку потрібно буде відредагувати. Детальніше на скріншотах :

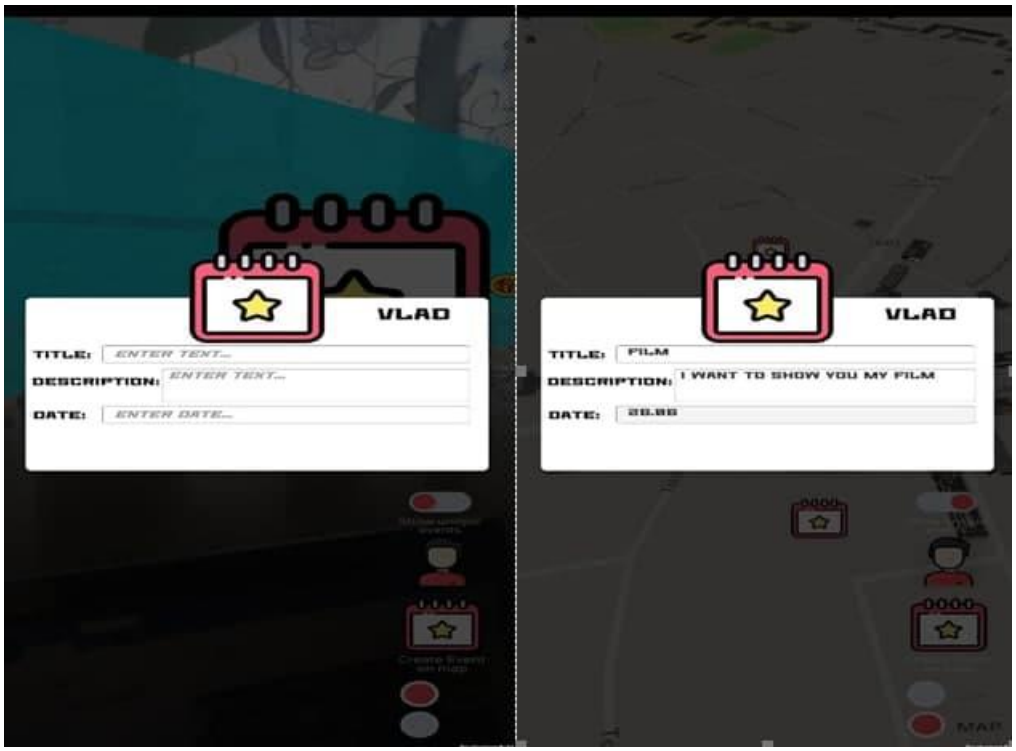


Рис. 4.3. Створення та редагування івенту

4.2 Висновок до розділу

Результат виконання розділу – це створення користувацького інтерфейсу, аналіз оптимального вигляду всіх вузлів програми, проведення змін та аналіз конкурентних продуктів, для оптимального розташування всіх вузлів програми.

ВИСНОВКИ

Під час виконання роботи було досліджено багато наукових досліджень, що розроблялись роками. Проаналізовано соціологічні результати у декількох розвинених країнах. Встановлено, що вільний час більшість людей проводять бездумно і без користі. Проблема проводження свого часу стосується більшості громадян нашої країни також. Таким чином, було поставлене завдання - запропонувати метод вирішення даної проблеми, який би зміг покращити культуру проведення вільного часу людиною. Як результат, після закінчення всіх робіт був розроблений прототип програмного продукту, що містить важливі функції, які б могли допомогти провести свій час з користю. Основні функції програми – це в зручному форматі відобразити найближчі місця, де можна провести час, також показати юзеру заходи, що створені іншими користувачами і надати зручну можливість створити свій власний. Ця ідея зможе послугувати розвитком ідей особистості та допомогти у починанні свого проекту або розвитку навичок комунікацій.

Перед початком створення програмного продукту було проаналізовано можливості створення програм під мобільні платформи. Були порівняні конкурентні платформи для розробки та свої навчики у програмуванні. Як результат було обрано основну платформу .Net та двигун Unity. На базі цих продуктів створено програмне забезпечення представлене у роботі. Для популярних фішок і залучення користувача було використано трендову технологію доповненої реальності – ARCore. Вона значно впливає на роботу програми і дозволяє користувачу обирати зручний режим використання додатку. Для орієнтування на місцевості та зображення мапи обрано MapBox, що є деталізованим та доволі зручним у використанні.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Опис технологій для реалізації мапи у додатку. – [Електронний ресурс]. Режим доступу: https://rapidapi.com/blog/top-map-apis/

2. Smartphone addiction facts & statistics. – [Електронний ресурс]. Режим доступу: https://bagby.co/blogs/digital-wellbeing-pills/smartphone-addiction-facts-statistics-updated-2020

3. Вільний час підлітків та молоді. – [Електронний ресурс]. Режим доступу: https://ukraine.ureport.in/opinion/2214/

4. Unity User Manual. – [Електронний ресурс]. Режим доступу: https://docs.unity3d.com/ru/current/Manual/UnityManual.html

5. Google AR – [Електронний ресурс]. Режим доступу: https://developers.google.com/ar/discover

6. Дослідження проведення вільного часу в Америці – [Електронний ресурс]. Режим доступу: https://deloitte.wsj.com/cmo/2019/01/03/work-sleep-tv-how-americans-spend-their-days/

7. Переваги та порівняння ARKit – [Електронний ресурс]. Режим доступу: https://apptractor.ru/info/articles/pochemu-arkit-luchshe-alternativ.html

8. Документація Mapbox – [Електронний ресурс]. Режим доступу: https://www.mapbox.com/

9. Гугл мапа – [Електронний ресурс]. Режим доступу: https://www.google.com.ua/maps/@50.2281449,30.3860612,16.21z?hl=ru

10. HERE мапа https://mobile.here.com/?x=ep

11. Порівняння MapBox – [Електронний ресурс]. Режим доступу: https://yalantis.com/blog/mapbox-maps-ready-mobile-apps/

12. Порівняння методів розробки додатку – [Електронний ресурс]. Режим доступу: https://app-android.ru/blog/na-chyom-razrabatyivat-prilozhenie-unity-3d-ili-nativno

ДОДАТКИ

Додаток А - Код, що описує кожен із префабів

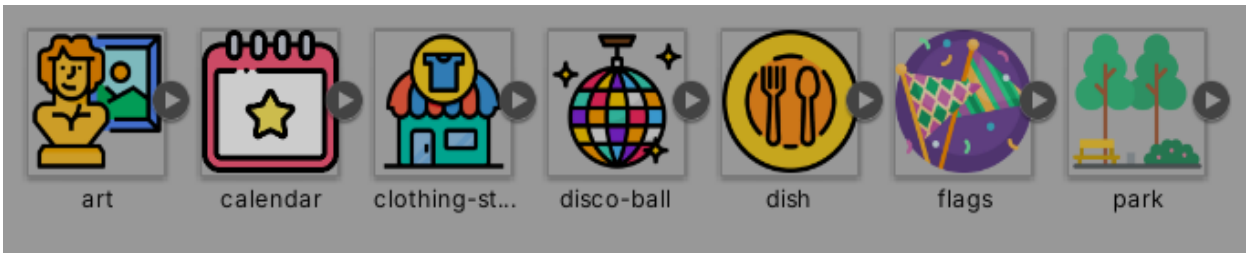
```
public class MyPoi : MonoBehaviour, IFeaturePropertySettable {
    public class PoiInfo {
        public Sprite poiSprite;
        public Sprite makiSprite;
        public string type = "";
        public string name = "";
    }
    public PoiInfo myInfo = new PoiInfo();
    public void Set(Dictionary<string, object> props) {
        myInfo.poiSprite =
transform.GetChild(0).GetComponent<SpriteRenderer>().sprite;
        if (props.ContainsKey("name")) {
            myInfo.name = props["name"].ToString();
        }
        if (props.ContainsKey("type")) {
            myInfo.type = props["type"].ToString();
        }
        if (props.ContainsKey("maki")) {
            myInfo.makiSprite = Resources.Load<Sprite>("maki/" +
props["maki"].ToString() + "-15");
        }
    }
}
```

Додаток Б - Код, що моделює поведінку Raycaster

```
public class Raycaster : MonoBehaviour {
    public LayerMask layerMask;
    public MyPopUp popUp;
    Ray _ray;
    RaycastHit _hit;

    bool IsUiMode {
        get { return popUp.IsOpened(); }
    }
    void Update() {
        if (IsUiMode) {
            return;
        }
        Camera cam = CameraManager.I.GetCurrentCamera();
        _ray = cam.ScreenPointToRay(Input.mousePosition);
        if (Input.GetMouseButtonDown(0)) {
            if (Physics.Raycast(_ray, out _hit, Mathf.Infinity, layerMask)) {
                MyPoi myPoi = _hit.transform.parent.GetComponent<MyPoi>();
                if (myPoi != null) {
                    popUp.Open(myPoi.myInfo);
                }
            }
        }
    }
}
```

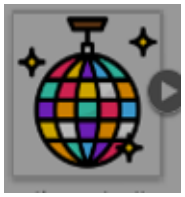
Додаток В – Опис функціональності точок розваг

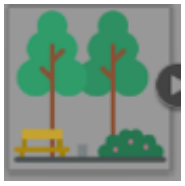


 - Галереї, мистецькі заклади.

 - Створені користувачем події.

 - Заклади харчування.

 - Клуби, дискотеки.

 - Парки, зелені зони.

 - Магазини одягу.

Додаток Г – Візуальне порівняння деталізації карт

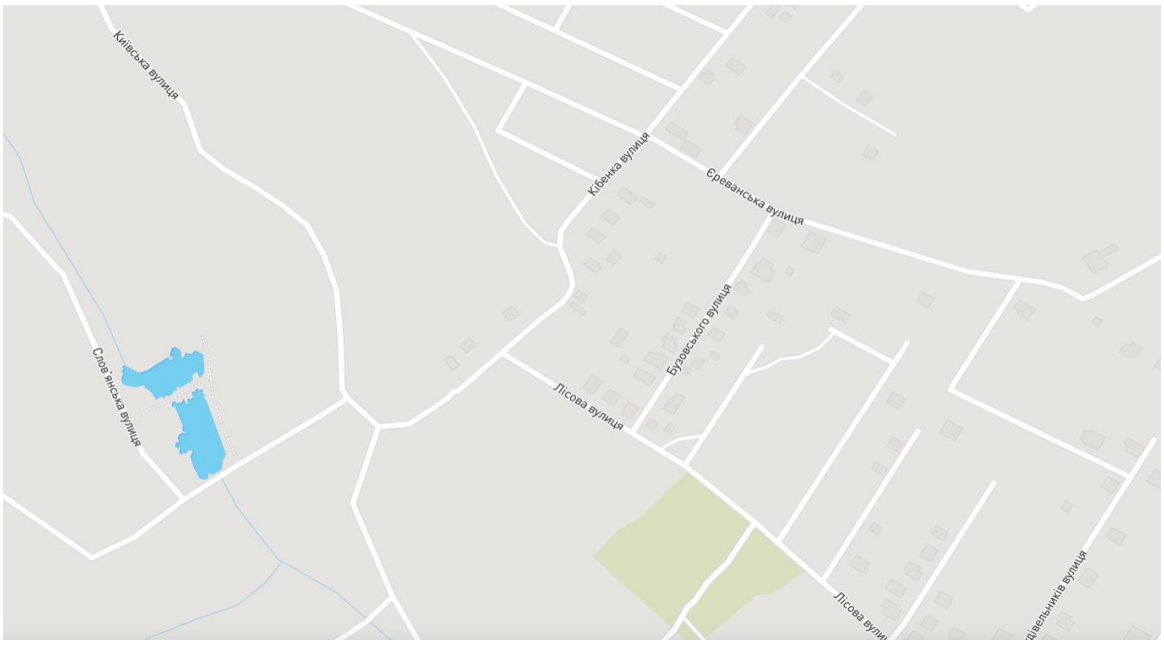


Рис. Г . 1 – Вигляд карти Mapbox

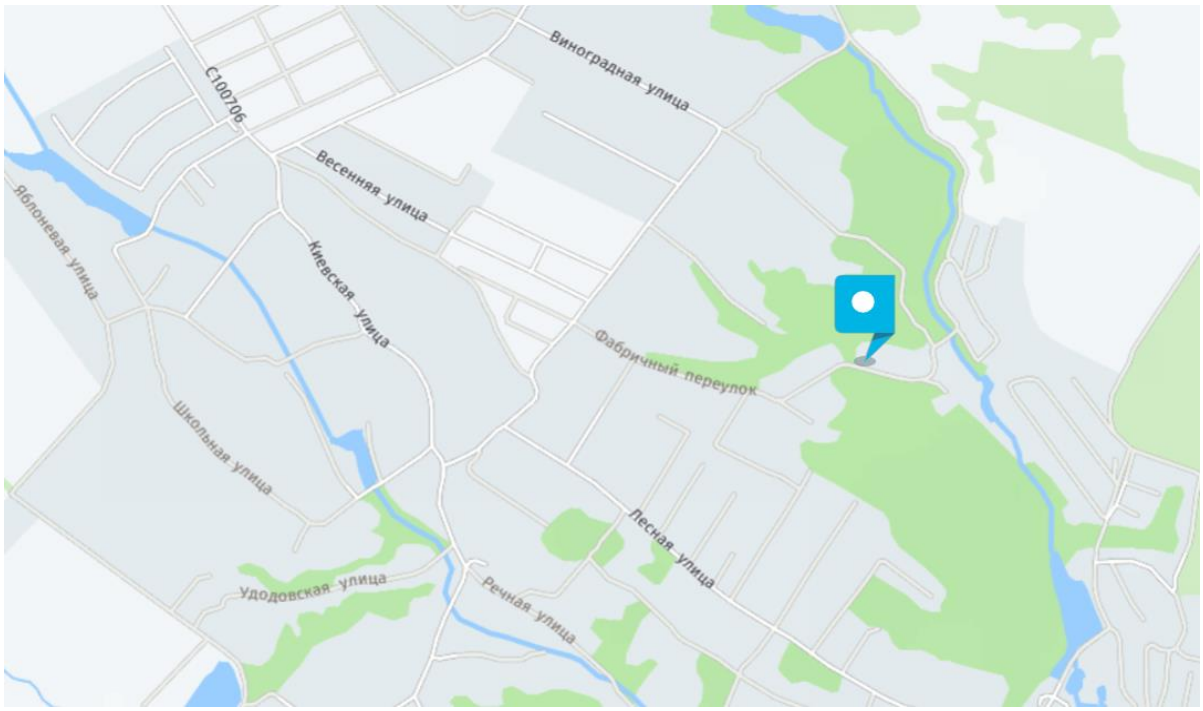


Рис. Г .2 – Вигляд карти HERE

Рис. Г. 3 - Вигляд карти Google Maps

Додаток Д – Software Architecture Document

Software Architecture Document

Entertainment search application using Unity and augmented reality

Vlad Fesiuk

Version 1.7

22/05/2021

Version	Description of Versions / Changes	Responsible Party	Date
1.0	Initial version	Vlad Fesiuk	17.04.2021
1.1	Start of functional view	Vlad Fesiuk	24.04.2021
1.2	Add the 3D view	Vlad Fesiuk	30.04.2021
1.3	-Added Entity Data Model dia -Added creating route.	Vlad Fesiuk	05.05.2021
1.4	- Added creating own POI	Vlad Fesiuk	11.05.2021
1.5	- Added new users	Vlad Fesiuk	15.05.2021
1.6	- Interface modified	Vlad Fesiuk	20.05.2021
1.7	Added Use Case View and Deployment View Diagram	Vlad Fesiuk	25.05.2021

Table of Contents

1. Documentation Roadmap

The Documentation Roadmap should be the first place a new reader of the SAD begins. But for new and returning readers, it is intended to describe how the SAD is organized so that a reader with specific interests who does not wish to read the SAD cover-to-cover can find desired information quickly and directly.

Sub-sections of Section 1 include the following.

- Section 1.1 ("Document Management and Configuration Control Information") explains revision history. This tells you if you're looking at the correct version of the SAD.

- Section 1.2 ("Purpose and Scope of the SAD") explains the purpose and scope of the SAD, and indicates what information is and is not included. This tells you if the information you're seeking is likely to be in this document.

- Section 1.3 ("How the SAD Is Organized") explains the information that is found in each section of the SAD. This tells you what section(s) in this SAD are most likely to contain the information you seek.

- Section 1.4 ("Stakeholder Representation") explains the stakeholders for which the SAD has been particularly aimed. This tells you how you might use the SAD to do your job.

- Section 1.5 ("Viewpoint Definitions") explains the *viewpoints* (as defined by IEEE Standard 1471-2000) used in this SAD. For each viewpoint defined in Section 1.3, there is a corresponding view defined in Section 3 ("Views"). This tells you how the architectural information has been partitioned, and what views are most likely to contain the information you seek.

- Section 1.6 ("How a View is Documented") explains the standard organization used to document architectural views in this SAD. This tells you what section within a view you should read in order to find the information you seek.

1.1. Document Management and Configuration Control Information

- Revision Number: *<<1>>*
- Revision Release Date: *<<22.05.2021>>*
- Purpose of Revision: *<<Find out the main architecture idea of the project>>*
- Scope of Revision: *<<list sections or page numbers that have been revised; provide a summary overview of the differences between this release and the previous one.>>*

1.2. Purpose and Scope of the SAD

The idea of this SAD is to explain the architecture of the project.

The main scope of this project is to implement this application in everyday life to help people with spending free time and easier implementation of ideas.

This SAD specifies the software architecture for < Entertainment search application using Unity and augmented reality>**.** All information regarding the software architecture may be found in this document.

1.2.1 What is software architecture?

Model View Controller is the main architecture of the project. Program is devided on three levels.

1) Model.
2) View.
3) Controller.

Elements and relationships.

The program interacts with external resources using api and aggregation relations. The software architecture first and foremost embodies information about how the elements relate

to each other. This means that architecture specifically omits certain information about elements that does not pertain to their interaction. Thus, a software architecture is an *abstraction* of a system that suppresses details of elements that do not affect how they use.

Multiple structures.

Although software architecture tends to focus on structural information, behavior of each element is part of the software architecture insofar as that behavior can be observed or discerned from the point of view of another element. This behavior is what allows elements to interact with each other, which is clearly part of the software architecture and will be documented in the SAD as such. Behavior is documented in the element catalog of each view.

1.3. Viewpoint Definitions

To program the application, two different methodologies for building software were considered:

1) Creating a product using a native environment. This meant writing for the Android platform while maintaining performance and detailed development in Java.

2) Creating a mini-game using the Unity game engine and the .Net platform.

The advantages and disadvantages of each option were analyzed.

1.3.1. Native development

Advantages of native development:

• Productivity. This point should be especially considered for mobile versions. This technology uses the battery sparingly, has an increased speed of processes. All this is possible thanks to developments for individual platforms. Therefore, this method allows you to effectively use applications.

• Interface. Unlike the first "logic" of creation, the code is written under the concrete system therefore visualization is accurately adjusted under the user, all "buttons" settle down in a usual place.

• Access to utilities. There are no restrictions on the use of the same geolocation or Bluetooth, which allows you to solve specific problems.

• Testing. It is easier to test a native program or game. This process uses all the mechanisms that easily "read" errors and inaccuracies. They, as a result, are removed by developers. The system update on platforms also includes the ability to dispose of problems for a specific user already in the process of use.

• Mode. The application, developed on a native basis, works both online and without the Internet.

• Store support. By downloading the application from the App Store or Google Play, the user can be sure of the quality, because at the slightest discrepancy (graphics, sound, controls, visual content) the program is simply not allowed into the store, which positively affects the reputation of stores and similar technology. Moreover, downloading programs from an official source is a guarantee of the absence of any interruptions and viruses.

Disadvantages of native development:

• Spending money and time. Entire teams can work on creating one application. This is easy to explain, because for each version, the operating system needs a specialist who needs to be paid. As a result, the budget increases and a lot of time is lost. The latter factor should also be taken into account and prepared in advance. This is especially important to understand when updating and supporting - not one application is created, but at least two. So, there should be more professionals as well.

• Lack of compatibility. This condition is a common consequence, as code is written for a specific OS that cannot be integrated and run on another platform. For example, if the application is designed for Android, it will be incompatible with iOS.

• Reduction of income. This fact is quite real, because the game or a useful calendar is written only on one platform. If in the future there is no reference to other systems, then part of the market is lost in any case, and therefore decreases profits.

The advantages and disadvantages of developing a native product were described above. Next, let's take a closer look at Unity.

1.3.2. Unity

Pros of using Unity:

• Easy to use. This includes the language itself, which is multilevel. The programmer does not need to invent something again, because the language is already implemented. The developer only needs to add others to one object. For example, if you create the main character of the game, then he needs control and an outer shell.

• Budget development. The budget becomes much smaller, thanks to the same platform. The code works on all devices at once, therefore, the number of developers and working hours decreases. Accordingly, the salary decreases. The engine itself is free. Of course, there is a paid subscription, but for most it may not be needed.

• Time. Based on the same "intersection", the time to create an object is reduced. The lack of unique interface elements and a single technology also contributes to the reduction of hours.

• Support. This technology has one of the strongest communities. The official website contains all the information on each step of development. If you have a question and it seems that there is no answer, you should go to the site and immediately understand what's going on. Even if there is no specific example, no one has canceled the support service.

• Asset Store. In this place are all the necessary resources to create a program. Convenient search and sorting allow you to quickly find what you need.

• The only logic. One technology allows, firstly, to perform updates simultaneously on all devices, and secondly, to avoid duplication, as a single system contains fewer errors.

Disadvantages of Unity 3D

Nothing can be perfect, so there are some disadvantages.

• Slowness. This factor often stems from applications that involve scale, complex scenes. You have to use additional utilities, and all this affects performance.

• "Weight". Almost all programs have a large volume. When it comes to a mobile app, it can take a lot memory in the phone. Even if you include a minimum in the simple game, ie sound, graphics, the installation file will weigh about 20 MB.

• Lack of "independence". It is that when you use Unity, their logo is automatically embedded. You can get rid of it by paying the PRO version.

• Restrictions. It is the lack of control functions for "sewn" parameters, from the camera to the database. This makes the application "inflexible" and eliminates work at maximum capacity.

Thus, after analyzing the two ways of creating software, the main advantages of each method were identified, compared and the game engine Unity was chosen. The main criterion was the cross-platform nature of the product, because it will significantly affect the development time.

The main development environment will be Unity.

The Unity development environment allows you to use a variety of additional libraries, ready-made assemblies to improve your product. Opens to the user the boundless world of the created graphics which can be added to the software and to make it as close as possible to game with various effects and functionality.

1.3.2.1. Abstract

Laisure time is an integral part of organizing the life of society and a person. Rational use of free time is a combination of types of activities, their active and passive forms, which effectively affects the individual, the development of its essence, physical, emotional, and intellectual spheres of its life. That is why was decided to create an application where a person will have an opportunity to find an activity or create own event.

2. Architecture Background

2.1. Problem Background

Main problem was to find an optimal tools for creating the program.

2.1.1. System Overview

I have used MVC pattern architecture for this project. It is divided on model, view, controller parts.

2.1.2. Goals and Context

There are some key requirements and system constraints that have a significant bearing on the architecture.

They are:

1. The system is built as a concept of the program in the future, which would prove the efficiency of the idea and show the possibilities of its implementation in the modern world. Thus, the documentation is intended for programmers who will refine the system.

2. The system will be written using Unity technologies with .NET platform.

3. The system must communicate with MapBox API. Defining how the system interfaces with these third-party systems is a primary concern of the architecture.

4. We will use ARCore plugin with Unity.

2.2. Solution Background

Non-functional requirements

1. High performance: The system must be able to receive a big number of users and be able to process their changes and store it on the database.

2. User friendly.

3. Security: all personal data must be totally secured, in order to prevent leaks.

4. Human errors: humans are the #1 source of involuntary (or voluntary) cause of problems in the systems. The system should always check the user input and, in general, any instruction.

5. Changeability: everything is very likely to change, so the system must be able to handle any kind of changes in features.

Use-Case View

The purpose of the use-case view is to give additional context surrounding the usage of the system and the interactions between its components. For the purposes of this document, each component is considered a use-case actor. Section 5.1 lists the current actors and gives a brief description of each in the overall use context of the system. The most common use-cases are outlined and illustrated using UML use-case diagrams and sequence diagrams to clarify the interactions between components.

Actors

User

The user will drive all operation of the software. The user interacts with all available interfaces to initiate and monitor all application operations.

Use-Case Realization

The main part of users` abilities:

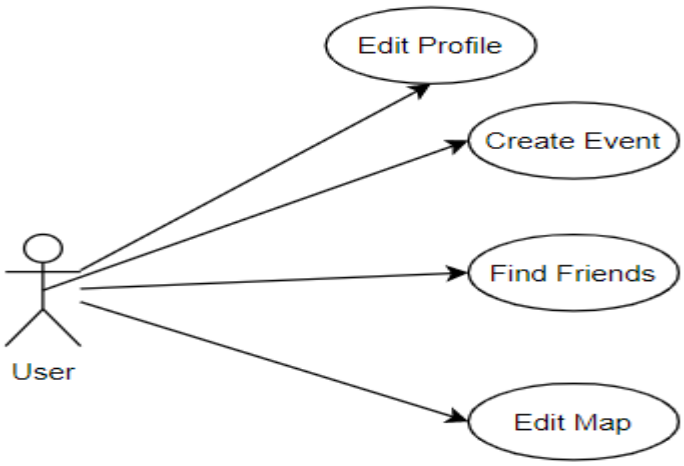
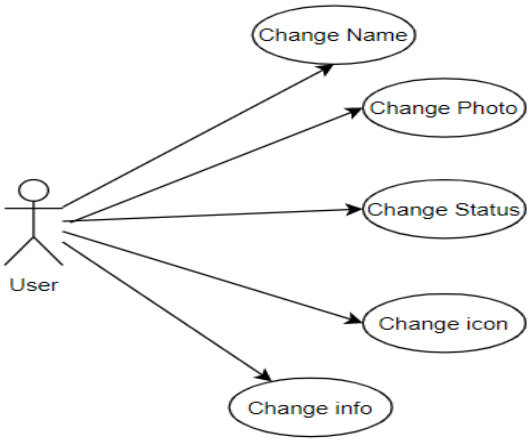


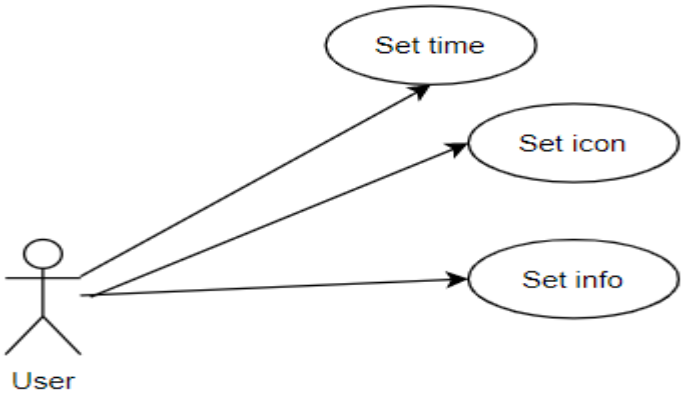
Figure 5.1 Possible uses of application

Let`s see detailed diagrams of each ability

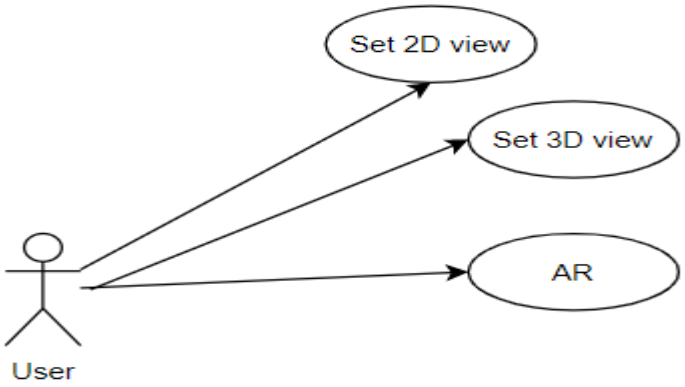
Edit Profile



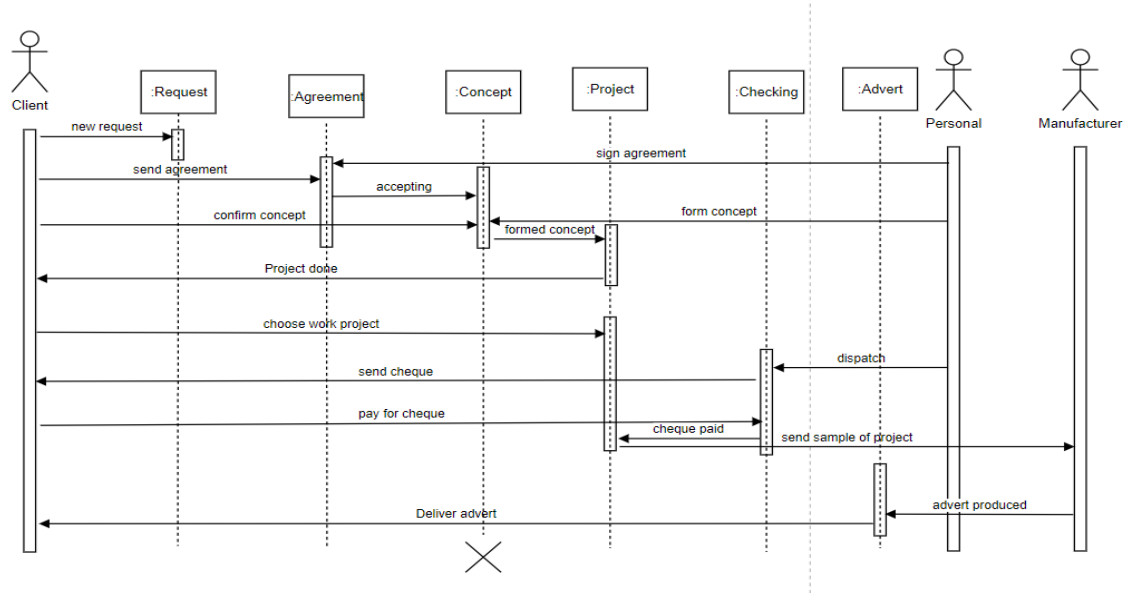
Create Event



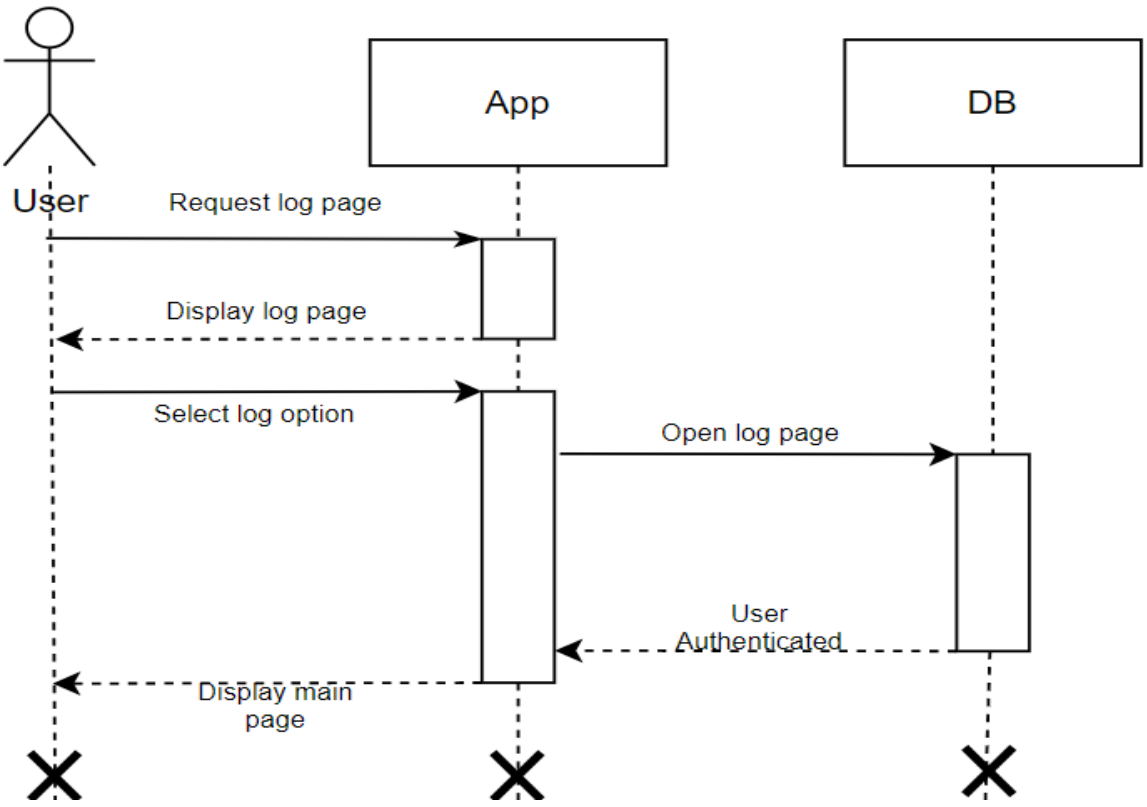
Set map



Sequence Diagram of adverts in the app



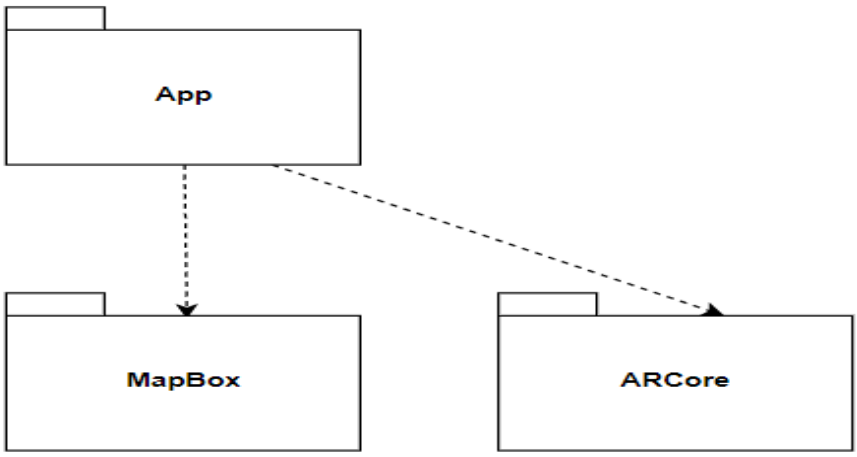
Sequence Diagram of login in the app



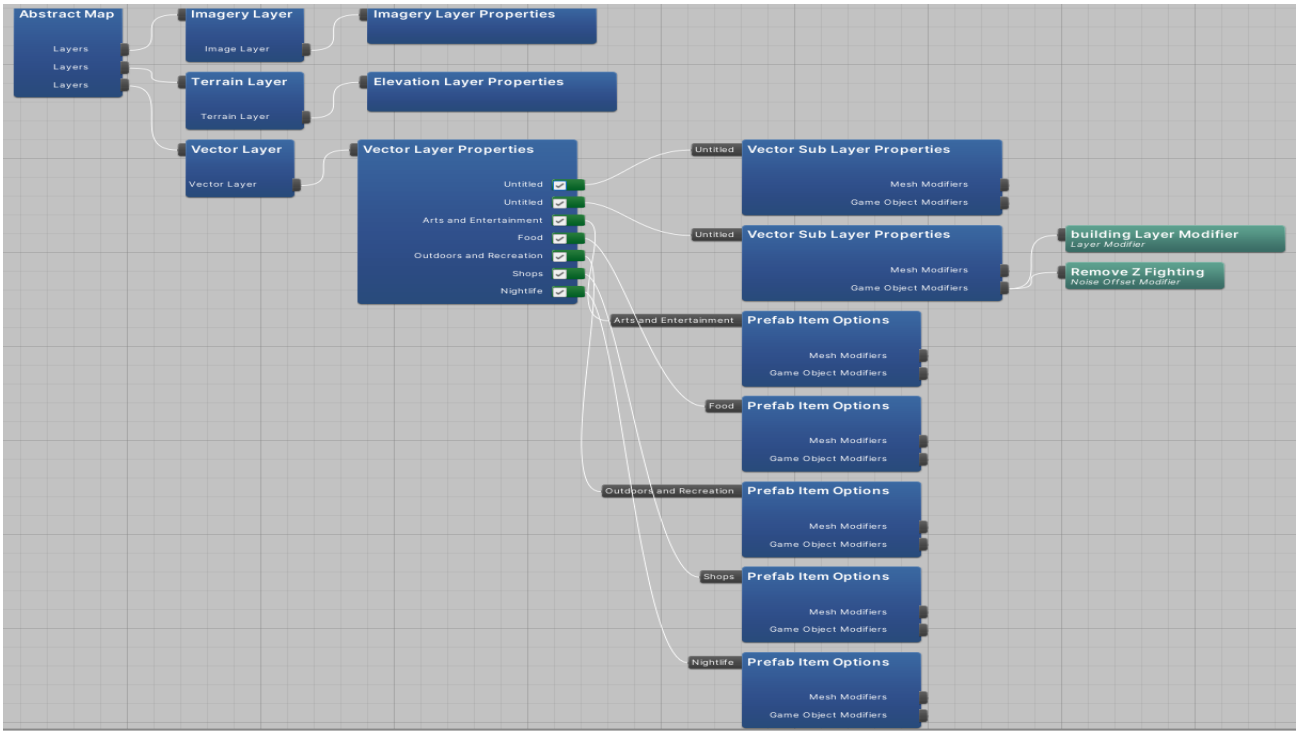
Components View

Components are objects that perform certain functions and have relationships with each other. Depending on the customer's requirements, components, their connections and quantity may vary. The component must have unique functionality.

High level of abstraction of the application



More about map` components



3. Referenced Materials

CONTENTS OF THIS SECTION: This section provides citations for each reference document. Provide enough information so that a reader of the SAD can be reasonably expected to locate the document.

Barbacci 2003	Barbacci, M.; Ellison, R.; Lattanze, A.; Stafford, J.; Weinstock, C.; & Wood, W. *Quality Attribute Workshops (QAWs)*, Third Edition (CMU/SEI-2003-TR-016). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. <http://www.sei.cmu.edu/publications/documents/03.reports/03tr016.html>.
Clements 2002	Clements, Bachmann, Bass, Garlan, Ivers, Little, Nord, Stafford, *Documenting Software Architectures: Views and Beyond*, Addison Wesley Longman, 2002.
IEEE 1471	ANSI/IEEE-1471-2000, *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*, 21 September 2000.

4. Directory

4.1. Glossary

4.2. Acronym List

UML – Unified Modeling Language

User - This is any user who is registered in the app.

API - Application Programming Interface.

OS - Operating System

SAD - Software Architecture Document

5. Conclusion

After describing the program architecture we have a detailed view of the developed software. The review presents the main points that require attention and are important during the development of the application.