Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерної інженерії та управління_____
(повна назва)

Кафедра ____Автоматизації проектування обчислювальної техніки_____
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти ___другий (магістерський)_____
(рівень вищої освіти)

Тривимірні моделі з елементами доповненої реальності в системі кіберуніверситету
(тема)

Виконав: студент 2 курсу, групи СКСм-19-1__
_____Янко А.Д._____
(прізвище, ініціали)
Спеціальність _____
____123 Комп'ютерна інженерія_____
(код і повна назва спеціальності)
Тип програми _____
____освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)
Освітня програма_____
__Спеціалізовані комп'ютерні системи_____
(повна назва освітньої програми)
Керівник _____доц. Шкіль О.С._____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ __Чумаченко С.В._____
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
Кафедра Автоматизації проектування обчислювальної техніки
Рівень вищої освіти другий (магістерський)
Спеціальність 123 – Комп'ютерна інженерія
Тип програми Освітньо-професійна
Освітня програма Спеціалізовані комп'ютерні системи

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____» _____ 2020 р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ (ПРОЕКТ)

Студентові Янко Антону Дмитровичу
(прізвище, ім'я, по батькові)

1.Тема роботи (проекту) Спеціалізований програмно-апаратний застосунок для тренування візуальної реакції
затверджена наказом по університету від " 30 " 10 2020 р. № 1489 Ст.
2. Термін подання студентом роботи (проекту) 16.12.2020
3. Вихідні дані до роботи (проекту)
Мобільний пристрій під керуванням операційної системи iOS
Мова програмування Swift
Середовище розробки XCode
360 камера для створення сферичних панорам
4. Зміст пояснювальної записки (перелік питань, що потрібно розробити):
Розглянути технології доповненої реальності
Розглянути технології сканування зображень
Ознайомлення з особливостями роботи з сервісом CIST
Ознайомлення з особливостями роботи з базою даних реального часу
Редагування панорамних фотографій та їх склейка у циліндричні
Створення віртуального туру з отриманих фотографій
Створення мобільного додатку
Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників, плакатів)
презентація

6. Консультанти розділів роботи (проекту)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

7. Дата видачі завдання 01.09.2020

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи (проекту)	Термін виконання етапів проекту (роботи)	Примітка
1	Отримання завдання на атестаційну роботу	01.09.2020	виконано
2	Пошук та аналіз літературних джерел за темою дипломної роботи	01.10.2020	виконано
3	Аналіз особливостей роботи з бібліотекою ARKit	10.10.2020	виконано
4	Створення панорамних фотографій	15.10.2020	виконано
5	Програмна реалізація	01.11.2020	виконано
6	Опис програмної реалізації	10.11.2020	виконано
7	Оформлення роботи	12.12.2020	виконано
8	Представлення роботи до захисту	16.12.2020	виконано

Студент _____ _____
(підпис)

Керівник роботи (проекту) _____ _____ доц. кафедри АПОТ Шкіль О.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до атестаційної роботи містить 106 сторінок, 14 рисунків, 11 джерел посилання.

3D-ПАНОРАМА, ЦИЛІНДРИЧНА ПАНОРАМА, ВІРТУАЛЬНИЙ ТУР, ПАНОРАМА, ПРОГРМНЕ ЗАБЕЗПЕЧЕНЯ, КОНТРОЛЬНІ ТОЧКИ, ДОПОВНЕНА РЕАЛЬНІСТЬ, МОБІЛЬНІ ТЕХНОЛОГІЇ, СКАНУВАННЯ ЗОБРАЖЕНЬ, ДІДЖИТАЛІЗАЦІЯ ОСВІТИ

Метою передатестаційної практики є аналіз галузей застосування віртуальних турів та додатків інших університетів, розробка додатку для кіберуніверситету та навігації по ньому, зйомка 3D-панорам для додатку.

Додаток кіберуніверситету може бути застосований студентами та їх батьками. Студенти можуть переглядати розклад, інформацію про керівників, та для пошуку необхідної аудиторії. Батьки зможуть оглянути аудиторії за допомого 3D-панорам, якщо аудиторія в даний час зачинена, а також переглянути повну інформацію про обладнання необхідної аудиторії.

Розроблено та протестовано мобільний додаток, що показує за допомогою камери смартфона інформацію про аудиторії та викладачів в графічному інтерфейсі додатку або в тривимірному просторі. Для реалізації мобільного додатку використана база даних реального часу – Firebase, сервер CIST та фрейморк ARKit.

Мобільний додаток реалізований на базі операційної системи iOS. Програмна реалізація алгоритму додатку здійснена на мові програмування Swift у середовищі розробки XCode.

ABSTRACT

The explanatory note to the attestation work contains 106 pages, 14 imags, 11 sources of reference.

3D-PANORAMA, CYLINDRICAL PANORAMA, VIRTUAL TOUR, PANORAMA, SOFTWARE, CHECK POINTS, AUGMENTED REALITY, MOBILE TECHNOLOGIES, IMAGE SCANNING, DIGITALIZATION OF EDUCATION

The purpose of pre-certification practice is to analyze the areas of application of virtual tours and applications of other universities, development of an application for cyber university and navigation on it, shooting 3D-panoramas for the application. The cyber university application can be used by students and their parents. Students can view schedules, information about supervisors, and search for the required audience. Parents will be able to view the audience with the help of 3D-panoramas, if the audience is currently closed, as well as view complete information about the equipment of the required audience. A mobile application has been developed and tested, which uses a smartphone camera to display information about audiences and teachers in the application's graphical interface or in three-dimensional space. A real-time database - Firebase, CIST server and ARKit framework - was used to implement the mobile application. The mobile application is implemented on the basis of the iOS operating system. The software implementation of the application algorithm is carried out in the Swift programming language in the XCode development vironment.

ЗМІСТ

SWIFT	–	Мова програмування
ARKit	–	Фрейворк для роботи з доповненою реальністю
iOS	–	iPhone operating system
ДР	–	Доповнена реальність
БД	–	База даних
СКБД	–	Система керування базами даних
XCode	–	Середовище розробки програмного забезпечення
IDE	–	Integrated Development Environment (інтегроване середовище розробки)
GUI	–	Graphical user interface (графічний інтерфейс користувача)
AR	–	Augmented reality (доповнена реальність)
3D	–	3-dimensional

ВСТУП

Завданням даної роботи є створення віртуального 3D туру з серії віртуальних фотопанорам. Елементами віртуального туру, як правило, є сферичні, циліндричні або кубічні панорами, які з’єднані між собою інтерактивними посиланнями-переходами (маркерами).

Віртуальний тур є ефективним інструментом маркетингу, що дозволяє показати потенційному споживачеві товар, послугу або об’єкт особливим чином. До переваг 3D панорамі, можна віднести те, що вона охоплює набагато більше простору і користувач може детально розглянути все навколо точки зйомки, оскільки кожна панорамна фотографія охоплює 360 градусів огляду. Перегляд 3D панорами створює ефект присутності в точці зйомки. На основі панорамних фото збираються пов’язані між собою переходи.

Додаток розроблений для університету також може запропонувати покращені можливості орієнтації першокурсників в учбовому процесі, а також мотивувати студентів на позаучбове життя та вивчення новітніх технологій. Смартфон зараз є у кожного студенту, та вже існує приклади додатків, які мають в собі розклад занять. Ця робота пропонує не лише розклад занять для студентів, як всі існуючі аналоги, але і відображення любої інформації про викладачів та аудиторії, за допомогою пошуку всередені додатку, або за допомогою спеціальних унікальних карток, які згенеровано спеціально для цього додатку. При наведені камери телефону на картку, студент побачить всю інформацію про викладача, або аудиторію, в доповненій реальності, а також зможе переглянути цю інформацію всередені додатку.

В рамках цієї роботи буде розглянутий процес створення та використання додатку. Використання такого роду додатку для отримання любої інформації про університет в тривимірному, або в двовимірному просторі продемонструє

абітурієнтам сучасний підхід університету до нових технології.

1 ДОПОВНЕНА РЕАЛЬНІСТЬ

1.1 Історія та розвиток доповненої реальності

По-справжньому широка публіка зіткнулася з доповненою реальністю, коли компанія Google створила свої розумні окуляри. Після цього була епоха масок, які робили з нас котиків, зайчиків. Потім гра Pokemon GO захопила обидві реальності і змусила намотувати кілометри. А в 2017 році компанія Apple презентувала ARKit, а Google - ARCore, і це значить що з’явиться нова хвиля ігор та додатків із застосуванням доповненої реальності, можливості якої набагато ширше і корисніше для суспільства, ніж ловля покемонів.

Доповнену реальність треба відрізняти від віртуальної і змішаної. У доповненої реальності віртуальні об'єкти проектуються на реальне оточення. Віртуальна реальність - це створений технічними засобами світ, який передається людині через органи чуття. Змішана або гібридна реальність об'єднує обидва підходи. Тобто, іншими словами, віртуальна реальність створює свій світ, куди може зануритися людина, а доповнена додає віртуальні елементи в світ реальний.

Історія доповненої реальності, бере початок з розробок віртуальної реальності. Батьком віртуальної реальності був Мортон Хейліг. 28 серпня 1962 року він запатентував симулятор Sensorama (рис. 1.1), за це він і отримав звання батька віртуальної реальності.

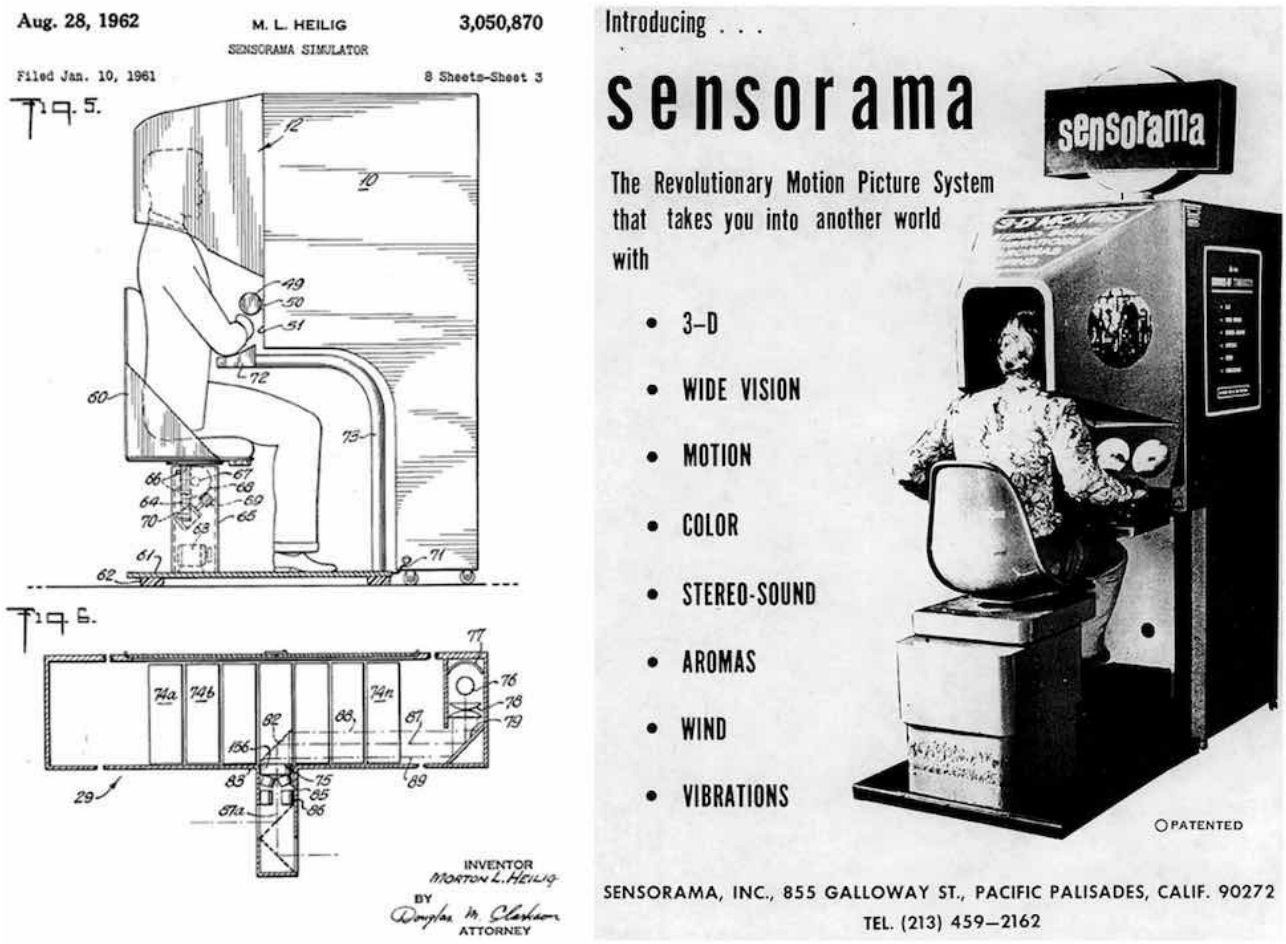

Рисунок 1.1 – Симулятор Sensorama

Це був пристрій ранньої версії віртуальної реальності, а не доповненої, але саме він дав поштовх до розвитку обох напрямків. Хейліг навіть винайшов спеціальну 3D-камеру, щоб знімати фільми для Сенсорами.

У 1968-му році комп'ютерний фахівець і професор Гарварда Айван Сазерленд зі своїм студентом Бобом Спрауллом розробили пристрій, що одержав назву «Дамоклів меч» (рис. 1.2). Це була перша система саме доповненої реальності на основі головного дисплею.

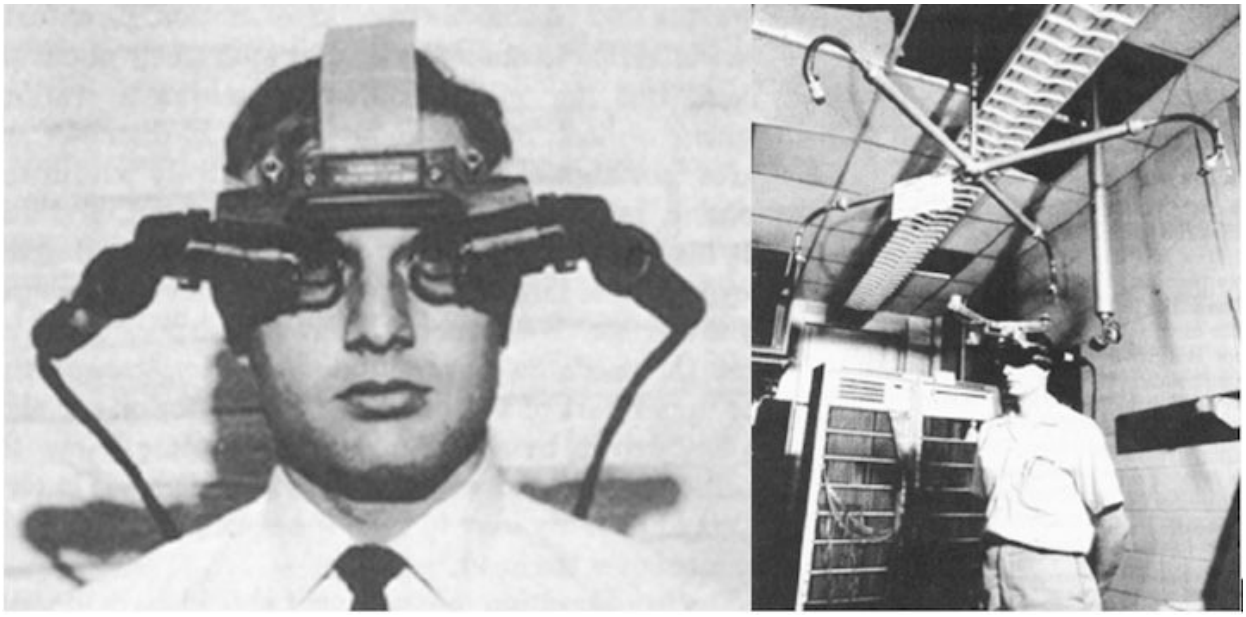

Рисунок 1.2 – Пристрій “Дамоклів меч”

Через десять років, в 1978-му, Стів Манн придумав перше пристосування для доповненої реальності, яке не було прикручене до стелі. У EyeTap використовувалася камера і дисплей, що доповнює середу в режимі реального часу. Цей винахід став основою для майбутніх проектів, але масово не використовувався.

Перше масове використання доповненої реальності стало можливе завдяки Дену Рейтону, який в 1982-му році використовував радар і камери в космосі для того, щоб показати рух повітряних мас, циклонів і вітрів в телевізійних прогнозах погоди.

У 90-ті пошук нових способів використання продовжився, а вчений Том Коделл вперше запропонував термін «доповнена реальність». Перед ним і його колегою поставили завдання: знизити витрати на дорогі діаграми, які використовували для розмітки заводських зон по збірці літаків Боїнг. І рішенням стала заміна фанерних знаків з позначеннями на спеціальні шоломи, які відображали інформацію для інженерів. Це дозволило не переписувати позначення кожного разу вручну, а просто змінювати їх в комп'ютерній програмі.

Далі розвиток відбувався стрімко. Стрибок, зроблений у виробництві мікропроцесорів, і, як наслідок, у всьому технологічним секторі, дозволив сильно прискорити роботи.

У 1993-му році в університеті штату Колумбія Стів Файнер представив систему KARMA (Knowledge-based Augmented Reality for Maintenance Assistance, перекладається приблизно як «Інтерактивний помічник для обслуговування»), яка дозволяла через шолом віртуальної реальності побачити інтерактивну інструкцію з обслуговування принтера (рис. 1.3).

Рисунок 1.3 – Інтерактивний помічник для обслуговування

У 95-му Джун Рекімото зібрав Navicam - прототип мобільного пристрою доповненої реальності, якою її зараз знають користувачі смартфонів. Navicam був переносним дисплеєм із закріпленою на зворотному боці камерою, відео потік якої оброблявся комп’ютером і, при виявленні кольоровий мітки, виводив на екран інформацію про об'єкт.

У 96-му році Джуном Рекімото і Южді Аятцука був розроблений Матричний Метод. Він описує реальні і віртуальні об'єкти за допомогою плоских міток на зразок QR-кодів. Це дозволяло вписувати віртуальні речі в реальний світ, просто за допомогою мітки. Наприклад, покласти на підлогу листок з кодом, навести на нього камерою і ось у вас в кімнаті стоїть динозавр.

У 98-му році НФЛ вперше використовувала доповнену реальність,

розроблену компанією Sport Vision, в прямій трансляції спортивних ігор. Під час матчів на картинку з камери, оглядово яка б показала ігрове поле, додавалися технічні лінії та інформація про рахунок. Про «чарівної жовтої лінії» є старий сюжет.

У 99-му НАСА застосувала систему доповненої реальності в приладовій панелі космічного апарату Ікс-38, який навчився відображати об’єкти на землі незалежно від погодних умов і реальної видимості.

На самому початку розвитку AR було зрозуміло, що її успіх буде залежати від того, наскільки зручно буде нашим очам.

Найгучнішою подією доповненої реальності останніх років стали окуляри Google Glass, з якими є невелика плутанина. Незважаючи на те, що саме вони багатьом першими приходять на думку, коли мова заходить про доповнену реальність, до неї ці окуляри відносини майже не мали. Віртуальне середовище практично не взаємодіяло з реальним.

Окуляри вміли робити фото та відео за командою користувача, з автоматичною відправкою в хмару. Окуляри запустили хвилю, давши зрозуміти іншим компаніям, що можна всерйоз братися за розробку пристроїв доповненої реальності.

Естафету тут же прийняла Майкрософт, через пару років завуальовано анонсувала (а в 2016-м і представила) окуляри змішаної реальності Hololens. Правда, тільки для розробників і журналістів. Продукт складний, його досі розробляють. Але в інтернеті багато захоплених оглядів, де люди діляться своїм досвідом взаємодії з віртуальним середовищем.

Hololens не вимагають підключення до іншого ПК або телефону. У окулярів чотири камери, за допомогою яких вони аналізують кімнату і поєднують віртуальні об'єкти з реальним світом.

Окуляри дозволяють практично повноцінно працювати з Windows 10, причому, назва «Windows» набуває нового сенсу: вікна системи легко вішаються на стіни на манер, власне, вікон. Окуляри запам'ятовують приміщення, тому, коли користувач повертається в ту ж саму кімнату, всі вікна

додатків і інші елементи змішаної реальності чекають його на своїх місцях.

Зараз існує близько десятка найбільш перспективних розробників і продуктів для доповненої реальності в форм-факторі окулярів: Vuzix, Sony, ODG, Solos.

Але один виробник підібрався найближче до того, що може бути не тільки технологічно, а й зручно. Це - компанія Magic Leap.

Запустили свій проект в 2010-му році в атмосфері абсолютної секретності, він вже через пару років зібрав інвестицій більш ніж на півмільярда доларів від таких гігантів як Google і Qualcomm. Ніхто за межами вузького кола інвесторів не знав, чим ця компанія залучила таку увагу і що у неї за продукт. Пізніше було офіційно оголошено: компанія працює над просунутою версією окулярів доповненої реальності, які на голову сильніше аналогів і, на відміну від інших виробників, в Magic Leap рівну увагу приділяють як апаратному забезпеченню, так і програмному забезпеченню і інтерфейсів. Незважаючи на те, що компанію більше цікавить індустрія розваг, ніж прикладне застосування, на сьогоднішній день вона є лідером в зручності для користувача інтерфейсів.

Але поки AR в основному зустрічається в телефонах. Це зручність, готова технічна база, широка поширеність пристроїв і простота написання ПО.

1.2 Сфери використання доповненої реальності

Заточені під фото для соціальних мереж додатки пропонують приблизно одні й ті ж функції: маски і 3D персонажі в простір. Тобто - розваги. Але все більше компаній розуміють важливість цієї ніші і представляють більш утилітарні програми:

AirMeasure - віртуальна рулетка, здатна визначати відстані і розміри в 3д-оточенні;

Google Translate вміє перекладати текст, який бачить камера, в реальному часі;

Sun Seeker допомагає побачити траєкторію сонця на місцевості в будь-який день року;

Google Sky Map допомагає дізнатися, які зірки зараз видно на небі.

Саме в мобільному сегменті зараз сконцентровані найцікавіші AR-стартапи для масового ринку.

1.2.1 Сфера розваг

Головна мобільна сфера, де себе знайшла Доповнена Реальність - це, звичайно ж, розваги.

Ви напевно грали в шутери від першої особи. Але ви коли-небудь замислювалися, що відображення кількості патронів, здоров'я і аптечок - це теж доповнена реальність, тільки для вашого персонажа?

На початку 2000-х вийшов AR-порт легендарної гри Quake. Він так і називався: ARQuake.

У наш же час можна і самому стати героєм шутеру. Наприклад, в грі Father.IO. Такі проекти з'являються все частіше.

У 2014-му вийшла гра Night Terrors, один з перших популярних фільмів жахів в доповненої реальності. Спробуйте його вночі в якомусь підвалі - не забудеться.

У 2016-му студія Nyantic випустила спадкоємицю своєї гри Ingress і найголовнішу AR-гру, ймовірно, на багато років вперед: Pokemon Go. Доповнена реальність, геотрекінг і популярна на весь всесвіт - все склалося настільки вдало, що Pokemon Go завантажили понад сто мільйонів чоловік. Гра швидко стала феноменом і почала збирати навколо себе скандали, в тому числі в Україні. Pokemon Go унікальна ще й тим, що змусила мільйони людей гуляти на свіжому повітрі.

Настільні ігри отримали нову форму завдяки технології. Такі компанії як Lego і Disney активно ведуть розробку ігор з використанням AR, а наміри до них приєднатися висловили практично всі великі виробники іграшок. Дослідницькі групи вже зайнялися збором даних про те, як маленькі діти

взаємодіють з іграми і додатками доповненої реальності, і яким чином це впливає на їх сприйняття реального світу. Можливо, в майбутньому найцікавіші ідеї з розвитку технології будуть звучати від тих, для кого ця сама технологія була просто частиною дитинства.

Саме розваги сьогодні розвивають дослідницьку базу доповненої реальності. А завдяки колосальним обсягами даних, добровільно переданих людьми компаніям-розробникам, технологія в зв'язці з машинним навчанням роблять кроки в бік більш серйозних областей.

1.2.2 Від розваг до реального життя

Довідкова інформація, оголошення та віртуальні покажчики обов'язково увійдуть в наш віртуальний простір. Віртуальний екскурсовод проведе нас по руїнах замку, та ще й покаже сценку, як саме цей замок розвалили, і яким він був до того. Соціальні функції, на зразок фільтра по статусу «в активному пошуку», допоможуть знайти другу половинку прямо в натовпі.

Та реклама. Ось вже яка сфера спить і бачить якнайшвидше впровадження доповненої реальності в повсякденне життя. А свіжість і новизна формату забезпечать приголомшуючий ефект. AR з'явилася навіть в друкованих виданнях. Наприклад, у випуску Есквайр 2009-го року потрібно було відсканувати обкладинку, і тоді на ній оживав Роберт Дауні молодший.

Ще раніше AR і друковані видання схрестила BMW, випустивши в декількох німецьких журналах рекламу моделі MINI, яка на екрані ставала тривимірною і дозволяла себе розглядати з усіх боків.

А обкладинки, до слова, є не тільки у журналів і книг. Для того, щоб з вами почала розмовляти етикетка пляшки, сьогодні не потрібно навіть пити.

Комерційні можливості доповненої реальності настільки великі, що складно окреслити межі. Навіть графіті не залишилося осторонь від AR-технологій.

AR може використовуватися для швидкої примірки в магазинах: ідея зайти в меблевий і тут же на тестовому стенді зібрати собі кімнату з меблями і

побутовою технікою, користуючись підказками по сполучуваності, напрошується сама собою.

Більш цікаву і корисну ідею втілив маркетинговий відділ IKEA ще в 2014-му. Приміряти меблі з каталогу прямо до інтер'єру своєї кімнати виявилося вкрай заманливо.

1.2.3 Сфера освіти

Технологія може зайняти ту нішу, яка в науковій фантастиці віддана голограмам. Тільки голограми будуть ще не скоро, а пристрої такі як Hololens технічно майже готові. Перспектива побачити в вузах, а після і школах, віртуальні інтерактивні ілюстрації, які можна розглянути з усіх боків, з якими можна взаємодіяти і тут же бачити результат своїх дослідів, представляється прекрасним далеко зі світлих фантазій про майбутнє. Навчання будь-яким інженерним спеціальностям може стати куди більш наочним і легким для розуміння.

1.2.4 Сфера медицини

Тут прямо очі розбігаються від можливостей. Крім максимально наочного навчання студентів медичних вузів, відразу представляється візуалізація даних прямо на пацієнта, замість розставлених навколо екранів. УЗД стане максимально наочним. Ну і майбутня мама буде щаслива отримати на телефон тривимірну дитинку, якого буде з радістю крутити і розглядати, вишукуючи схожість того з батьком і собою.

Але одна справа УЗД, яке не вимагає оперативного втручання, і інша - небезпечні для життя пацієнтів операції, де наочність може допомогти лікарю швидше реагувати і точніше працювати.

Наочну анатомію в доповненому просторі демонструє HoloAnatomy для Hololens, який якраз і про медицину, і про освіту. А заодно - і один із знакових додатків для компанії Microsoft.

Менш драматично, але не менш корисно - помічники для сліпих і глухих,

повідомляють про предмети і події навколо або показують субтитри.

Наприклад, стартап Aira одночасно пропонує нейромережевого помічника, що розпізнає і проговорює все, що бачить камера окулярів, і живого співробітника стартапу, що допоможе зорієнтуватися по тій же камері в особливо складній ситуації. Система прив'язана до додатка для смартфона. Користувач за передплатою отримує окуляри з камерою і можливість транслювати зображення з них між співробітниками підтримки. Але постійно телефонувати їм немає потреби: голосовий асистент розпізнає тексти і образи, перекриваючи безліч повсякденних міських завдань. Логічно, що в міру розвитку комп'ютерного зору надбудова з живими співробітниками буде все менш актуальна, але сьогодні це хороший компроміс з людських і комп'ютерних ресурсів.

1.2.5 Сфера військових технологій

І якщо системи наведення в бойових винищувачах, дронах і танках для армії - це сьогодні справа звичайна, тому що саме з ранніх систем доповненої реальності для льотчиків і росли інші військові проекти в цій галузі. Наприклад, просунуті системи доповненої реальності для піхоти, які будуть впроваджуватися вже через пару років.

В американській армії вже сьогодні використовується система HUD 1.0: сильно вдосконалений прилад нічного бачення, який також виконує функції тепловізора і проектує в монокль на шоломі покажчик, що показує куди потрапить куля за поточного стану стовбура.

Полегшені аналоги таких систем вже більше п'яти років доступні на ринку. Балістичний калькулятор від компанії TrackingPoint, фактично замінює снайперу або будь-якому охочому, напарника.

На черзі - HUD 3.0, який повинен вийти в наступному році. Він буде мати можливість накладати на реальну картинку повністю цифрові шари місцевості, моделі будівель, плани поверхів, позиції ворогів і навіть самих ворогів. А це вже заявка на здешевлення військових навчань. Військові ігри обходяться

державним бюджетам в колосальні суми щороку, а за допомогою систем доповненої реальності солдати зможуть тренуватися з умовним противником не залишаючи меж бази.

1.2.6 Майбутнє доповненої реальності

Доповнена реальність - це не тільки ігри і селфі з віртуальними масками. Це гігантська кількість можливостей для комерційного застосування, нові горизонти в освіті, промисловості, медицині, будівництві, торгівлі і навіть туризмі. І далі має бути тільки цікавіше.

Комерційний зростання AR разючий. Їй, на відміну від віртуальної реальності, необов'язково спиратися на спеціалізоване залізо і громіздкі пристрої. Технологія прекрасно працює на наймасовішому пристрої - смартфоні.

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

Наступний крок - масовий вихід AR із зони розваг і соціальних мереж в сектор інформаційної підтримки. Автовиробники починають випускати додатки-доповнення до призначених для користувача інструкціям, що допомагають власникам наочно вивчити свій автомобіль. Все більше виробників техніки починають випускати додатки для ремонтних майстерень, які допомагають майстрам орієнтуватися у внутрішньому устрої складних приладів. Amazon думає над тим, щоб полегшити життя покупцям: сподобалися кеди на перехожому - навів на того телефон і тут же замовив собі такі ж.

На сьогоднішній час навіть у технологічних гігантів немає ясної картини подальшого розвитку доповненої реальності. Це час безперервного народження ідей, знаходження несподіваних способів застосування і усвідомлення всієї потужності цієї фантастичної колись технології - доповненої реальності.

1.2.7 Постанова задачі дослідження.

Результатом атестаційної роботи було створення нового сервісу кіберуніверситету. На даному етапі ми маємо 12 сервісів кіберуніверситету (рис. 1.4)

Рисунок 1.4 – Сервіси кіберуніверситету

Дана розробка атестаційної роботи намагається створити сервіс для полегшення приймальної комісії в введенні в курс справи абітурієнтів. Також така розробка допомагає університету в часи епідемії, тому що батьки та абітурієнти можуть проглянути університет без необхідності його відвідування.

Додаток розроблений під платформу iOS, тому що на даний момент, компанія Apple найбільш успішна в роботі з технологіями доповненої реальності. Аналогом для платформи Android, використовується мова Kotlin та бібліотека ARCore, але вона працює на багато гірше, тому що компанія Google не акцентує увагу на роботі з доповненою реальністю.

Для розробки додатку під платформу iOS можна використовувати мову Objective-C та Swift. Для розробки додатку була обрана мова програмування Swift, тому що Objective-C більше не підтримується компанією Apple. Мова Swift більш досконала, оптимізована і підтримується компанією Apple.

Для реалізації віртуального туру і елементів доповненої реальності

використовуються бібліотеки ARKit та Vision, в розробленому додатку буде використовуватись бібліотека ARKit, тому що вона в собі містить більше функціоналу та більш оптимізована для роботи з трьох вимірними об’єктами.

2 БАЗА ДАНИХ РЕАЛЬНОГО ЧАСУ FIREBASE

База даних реального часу Firebase - це база даних, розміщена у хмарі. Дані зберігаються як JSON і синхронізуються в реальному часі з кожним підключеним клієнтом. Всі ваші клієнти використовують один екземпляр бази даних і автоматично отримують оновлення з новітніми даними.

2.1 Основні можливості бази даних реального часу Firebase

Для роботи в реальному часі замість типових HTTP-запитів, база даних Firebase використовує синхронізацію даних кожен раз, коли дані змінюються, будь-який підключений пристрій отримує це оновлення протягом мілісекунд.

При офлайн роботі додатки з використанням Firebase залишаються чуйними навіть у автономному режимі, оскільки Firebase зберігає дані на диску. Після відновлення підключення клієнтський пристрій отримує будь-які зміни, які він пропустив, синхронізуючи його з поточним станом серверу

База даних Firebase може бути доступна безпосередньо з мобільного пристрою або веб-браузера; немає потреби в сервері додатків. Безпека і перевірка даних доступні через Firebase. Правила безпеки бази даних, правила на основі виразів, які виконуються при читанні або запису даних.

Масштабування за допомогою бази даних Firebase, ви можете підтримувати потреби у даних вашого додатка в масштабі шляхом розбиття даних на декілька екземплярів бази даних в одному проекті Firebase. Оптимізуйте автентифікацію за допомогою перевірки автентичності Firebase у вашому проекті та перевірте автентичності користувачів у примірниках бази даних. Керуйте доступом до даних у кожній базі даних за допомогою спеціальних правил бази даних Firebase Realtime для кожного екземпляра бази даних.

2.2 Як працює база даних реального часу Firebase

База даних Firebase дозволяє створювати багаті, спільні програми, дозволяючи безпечний доступ до бази даних безпосередньо з коду на стороні клієнта. Дані зберігаються локально, і навіть в автономному режимі події в реальному часі продовжують надаватись, надаючи кінцевому користувачеві чуйний досвід. Коли пристрій відновлює з'єднання, база даних у реальному часі синхронізує зміни місцевих даних з віддаленими оновленнями, які відбувалися під час перебування клієнта в автономному режимі, автоматично об'єднавши будь-які конфлікти.

База даних реального часу надає гнучку мову правил на основі виразів, яка називаються Firebase Realtime Database Security Rules, щоб визначити, як ваші дані повинні бути структуровані і коли дані можуть бути прочитані або записані. При інтеграції з аутентифікацією Firebase розробники можуть визначити, хто має доступ до тих даних і як вони можуть отримати до них доступ.

База даних Realtime - це база даних NoSQL і має різні оптимізації і функціональність у порівнянні з реляційною базою даних. API баз даних реального часу розроблений, щоб дозволити операції, які можна виконувати швидко. Це дає змогу створити великий досвід роботи в реальному часі, який може обслуговувати мільйони користувачів без шкоди для реагування. У зв'язку з цим важливо подумати про те, як користувачам необхідно отримати доступ до даних, а потім відповідним чином структурувати їх.

2.3 Типи даних для зберігання

Cloud Firestore - це гнучка, масштабована база даних для мобільної і серверної розробки з Firebase і Google Cloud Platform.

Firebase Remote Config зберігає вказані парами ключ-значення розробника, щоб змінити поведінку та зовнішній вигляд вашої програми, не вимагаючи від користувачів завантажувати оновлення.

Firebase хостинг розміщує HTML, CSS і JavaScript для вашого веб-сайту, а також інші засоби, що надаються розробником, такі як графіка, шрифти та іконки.

Cloud Storage зберігає такі файли, як зображення, відео, аудіо, а також інші контенти, створені користувачами.

3 ТЕХНОЛОГІЯ СТВОРЕННЯ 3D ПАНОРАМ

Фотопанорами зазвичай створюються з декількох спеціально підготовлених фотографій, які потім «зшиваються» за допомогою різних програм в єдину панораму. У фотоапараті повинна бути передбачена функція фіксації експозиції - ручний режим установки витримки і діафрагми, а так само ручний режим установки балансу білого, завдяки чому фотографії не будуть відрізнятися один від одного яскравістю і контрастністю.

Одним з важливих моментів створення панорами є необхідність обертання камери навколо нодальної точки об'єктива. Нодальна точка - це точка в об'єктиві камери, де перетинаються промені світла, що йдуть до матриці. При обертанні камери навколо цієї точки відсутній паралакс об'єктів. Паралакс - зміщення об'єктів переднього плану щодо об'єктів заднього плану при повороті камери. Таке зміщення може викликати труднощі при зшиванні панорами.

Знімати кожний наступний кадр потрібно так, щоб він перекривав попередній приблизно на 30%. Більший відсоток перекриття означає кращу якість збірки готової фотопанорами, завдяки більшій кількості подібних об'єктів. Необхідно стежити, щоб лінія горизонту залишалася незмінною, контролювати це дозволяють бульбашкові рівні на штативі. Крім того, місця швів краще розташовувати на досить однотонних об'єктах. При зйомці важливо, щоб освітлення не змінювалося, в кадр не потрапляли рухомі об'єкти: автомобілі, хмари, люди, що коливаються від вітру дерева.

Використання автоматичного фокусування при зйомці 3D панорами може призвести до того, що окремі кадри сфокусуються по-різному, або, ще гірше, деякі кадри виявляться зовсім поза фокусом, тоді зібрати панораму буде неможливо.

Бажано проводити зйомку в форматі RAW, який забезпечує додаткові можливості по обробці кожного кадру, наприклад зміна експозиції.

Існують різні види панорам.

Площинна - це звичайна ширококутна фотографія, складена з декількох кадрів.

Циліндрична панорама (циклорама) - це панорама з можливість огляду на 360 градусів по горизонталі, але тільки на 180 градусів по вертикалі. При розгляданні такої фотопанорами глядач знаходиться на осі циліндра і озирається навколо (або обертає циліндр навколо себе).

Якщо кут огляду фотопанорами по горизонталі дорівнює 360 градусів, то циклорама краще площинної розгортки, так як не вимагає вертикального розриву.

Сферична панорама. Панорама проектується на середину сфери. Якщо кут огляду по горизонталі дорівнює 360 градусів, а по вертикалі дорівнює 180 градусів, то панорама заповнює внутрішню поверхню сфери цілком.

При розгляданні такої фотопанорами глядач знаходиться в центрі сфери і озирається навколо.

Сферична панорама дозволяє помістити необхідні елементи над спостерігачем або під ним (а на це ні площинна розгортка, ні циклорама, природно, не здатні).

Кубічна панорама. Панорама проектується на середину куба.

З точки зору глядача кубічна панорама має всі переваги сферичної панорами і в ідеалі повинна створювати у глядача майже досконале уявлення того, що він дивиться саме на поверхню сфери (не рахуючи природного розрізнення між сферою і кубом, створюваного бінокулярний зір людини).

Перевагою кубічної панорами у порівнянні зі сферичною є простота виготовлення, зберігання, транспортування, оскільки мати справу доводиться вже не з поверхнею складної форми, а тільки з шістьма гранями куба - плоскими і квадратними.

4 СТВОРЕННЯ 3D ПАНОРАМ ДЛЯ ВІРТУАЛЬНОГО ТУРУ

4.1 Зйомка

Умова - створення циліндричної панорами. При здійсненні зйомки фотоапарат був переведений в повністю ручний режим, який позначається символом Р. Штатив з фотоапаратом був встановлений в середині кімнати

Далі послідовно знімався кадр за кадром, ступінь перекриття сусідніх кадрів становила приблизно 30-40%. Так як умовою було створення циліндричної панорами - необхідно було зняти лише один ряд фотографій по горизонталі на 360 градусів в обох кімнатах.

4.2 Вибір і завантаження зображень

Після того, як були отримані зображення, можна переходити до склеювання їх в панораму. Роботу починали з завантаження фотографій призначених для склеювання в PTGui.

Відкриті в програмі знімки відобразяться у вигляді стрічки.

4.3 Автоматичне склеювання знімків

Далі натискаємо кнопку Align images, програма запустить свій алгоритм аналізу знімків, проаналізує всі вихідні кадри і в перекриваються областях сусідніх кадрів згенерує контрольні точки. Потім PTGui поєднає фотографії та проведе їх оптимізацію. Після автоматичного вирівнювання на екрані з'явиться нове вікно Panorama, в якому можна змінювати орієнтацію, як окремих частин панорами, так і всієї панорами цілком. Крім цього, пересуваючи бічний повзунок, можна обрізати зайві об'єкти, які не будуть використовуватися в

циліндричної панорамі.

Іноді програма може недостатньо точно визначити місця зшивання деяких знімків, тому необхідно, вручну виконати процедуру з'єднання окремих зображень, шляхом додавання контрольних точок на ці зображення.

4.4 Вказівка додаткових контрольних точок

Контрольні точки являють собою пари відміток на з'єднуються зображеннях, які позначають збігаються деталі на знімках. Чим точніше розташовані контрольні точки і чим більше буде їх число, тим правильніше буде складено шов між окремими зображеннями. Для управління контрольними точками фотографій слід перейти на вкладку Control Points. У двох вікнах показані об'єднуються знімки, на яких видно пари контрольних точок. Всі ці точки пронумеровані і виділені кольором.

Алгоритм програми недосконалий, тому іноді контрольні точки можуть визначатися недостатньо чітко. В цьому випадку потрібно натиснути правою кнопкою миші на проблемній точці і видалити невдалу позначку, вибравши команду Delete.

Після цього можна вручну проставити контрольні точки, клацаючи по зображенню на об'єктах присутніх на обох кадрах. Парну контрольну точку програма створить сама, залишиться лише простежити за правильністю її розташування і, в разі необхідності, пересунути її на правильну позицію. Таким чином, встановлюємо максимум контрольних точок, намагаючись, по можливості, розставляти їх по всій перекривається області кадрів, а не тільки в одному місці. Також дуже важливо розмішати контрольні точки з максимальною точністю.

4.5 Оптимізація

Для досконалого склеювання панорами, тобто склеювання без видимих швів», дистанція між парами контрольних точок повинна бути мінімальною. В процесі оптимізації розраховується, яким чином повинні трансформуватися і вирівнюватися окремі кадри панорами, щоб мінімізувати дистанцію між контрольними точками.

Для цього відкриємо вкладку Optimizer, перед цим натиснувши Advanced на вкладці Project Assistant для переходу в розширений режим. Зі списку корекції дисторсії лінзи вибираємо опцію «Heavy + lens shift». Вибираємо алгоритм оптимізації PTGui (варіант з алгоритмом Panorama Tools дає результат краще рідного оптимізатора PTGui, але він недоступний в даній збірці, проте його можна завантажити окремо з сайту розробників). Запускаємо процес оптимізації, після чого з'явиться вікно з результатами, де вказана середня, мінімальна і максимальна дистанція між контрольними точками. Необхідно знизити на мінімум середню дистанцію між контрольними точками. У результатах оптимізації крім цифр PTGui дає ще одну оцінку проведеного процесу - "very bad", "bad", "not so bad", "not so good", "good", "very good" або "too good to be true" . Але орієнтуватися тільки за цими оцінками не варто, оскільки вони не враховують розмір вихідних фотографій.

Якщо отримали середню дистанцію більше 5px, то бажано її зменшити. Для цього необхідно підтвердити результат оптимізації і знову перейти до таблиці контрольних точок. Перевіряємо, щоб контрольні точки були впорядковані по дистанції. Якщо бачимо кілька контрольних точок з сильним відхиленням від середнього значення дистанції, то ці точки слід видалити. Після цього знову виконуємо оптимізацію і перевіряємо результат. Якщо значення залишилися високими, то слід повторити видалення контрольних точок з найгіршим значенням дистанції і оптимізацію, поки результат нас не влаштує. Але при цьому необхідно стежити, щоб залишилася достатня кількість контрольних точок для зшиваючи панорами і не з'являлися дефекти.

4.6 Створення панорами

Після завершення оптимізації переходимо на закладку Create Panorama. Тут можемо вибрати бажаний розмір, формат готової панорами, ім'я файлу і шлях для його збереження. PTGui Pro дозволяє зберігати панораму також у вигляді окремих шарів, де кожен шар відповідає кожному вихідного кадру панорами. Ця опція буває особливо корисною, якщо потрібно ретушувати на панорамі повторювані рухомі предмети.

Нарешті запускаємо процес складання панорами і чекаємо її завершення. Залежно від кількості і розміру вихідних фотографій, розміру результуючої панорами процес може тривати від кількох секунд до кількох годин. Так само, час очікування може залежати від обчислювальної потужності комп'ютера.

В результаті нашої роботи отримуємо проекцію циліндричної панорами.

4.7 Виправлення дрібних дефектів

На попередньому етапі була отримана циліндрична проекція. У різних областях є дрібні дефекти після складання знімків в панораму. Після додавання контрольних точок вручну результат покращився, однак все ж потрібно додаткове редагування в Adobe Photoshop.

При редагуванні використовується Штамп, Пензель відновлення, Трансформування. В результаті виправляються різні дрібні дефекти зображення, що залишилися після склеювання панорами.

5 ОБ’ЄДНАННЯ ПАНОРАМ В ВІРТУАЛЬНИЙ ТУР

Віртуальний тур - спосіб реалістичного відображення тривимірного багатоелементного простору на екрані. Елементами віртуального туру, як правило, є сферичні панорами, з'єднані між собою інтерактивними посиланнями-переходами (хотспотами). В віртуальні тури також включають циліндричні панорами, віртуальні 3D-об'єкти, звичайні фотографії, відео, звук і т. д.

Часто панорами в віртуальному турі мають прив'язку до карти за координатами місця зйомки і орієнтовані по сторонах світу.

Іншими словами, віртуальний тур є загальним позначенням для декількох сферичних панорам, пов'язаних між собою за допомогою точок переходу, за якими в процесі перегляду можна віртуально «переміщатися». В віртуальні тури, як правило, включають і інші інтерактивні елементи: спливаючі інформаційні вікна, пояснюючі написи, графічно оформлені клавіші управління і т. д.

Віртуальний тур є ефективним інструментом маркетингу, що дозволяє показати потенційному споживачеві товар або послугу особливим чином. Він створює у глядача «ефект присутності» - яскраві, що запам'ятовуються зорові образи, і дозволяє отримати найбільш повну інформацію про товар або послугу.

5.1 Елементи віртуального туру

Відмінною особливістю віртуального туру є можливість переміщення від панорами до панорами. Вона реалізується за допомогою, так званих, точок переходу, які дозволяють «подорожувати» по віртуальному туру в залежності від бажання глядача. Щоб зробити віртуальний тур зручнішим і інформативним

використовується безлічі елементів: гаряча точка (hotspot), кнопки і панель управління, інтерактивний список панорам і мініатюри, карта віртуального туру і радар, спливаючі вікна з текстом, відео, звуковий супровід.

Гаряча точка - це зображення, текст або область панорами, які є посиланням. При наведенні курсора на гарячу точку спливає підказка, а натискання активує завантаження відповідної 3D панорами або відкриває спливаюче вікно. Гарячих точок на одній панорамі може бути кілька, вони логічно відповідають напрямку руху, яке ми використовуємо в реальності. Тобто в приміщенні вони, швидше за все, будуть розташовані на дверях, а на відкритій місцевості позначать точку проведення фото зйомки.

Кнопка і панель управління. Кнопка це основний елемент управління віртуальним туром. У вигляді кнопки можна використовувати текст, графічне зображення у форматі jpg, bmp, gif, png або swf-файл. Кнопокам привласнити кілька значень одночасно, якщо вони не суперечать один одному. Найчастіше використовуються наступні значення: управління рухом панорами (Включення / Відключення обертання, Збільшення / Зменшення зображення, скидання, і т.п.); перемикання між панорамами; включення / Відключення повноекранного режиму; включення / Відключення звукового супроводу; відкриття / закриття спливаючого вікна; відкриття посилання, відправка E-mail, відправка на друк; панеллю керування називають групу статичних кнопок розміщених по периметру вікна; інтерактивний список панорам і мініатюри.

Інтерактивний список включає в себе найменування всіх 3D панорам. Він дозволяє швидко знайти і відкрити будь-яку панораму, що входить до складу віртуальної екскурсії. Мініатюри представляють собою той же список, тільки оформлений у вигляді фотографій.

Карта віртуального туру дозволяє полегшити орієнтацію в просторі цифровий реальності. А гарячі точки, розміщені на карті, мають додаткову функцію - радар. Радаром називають кольорову зону, що обертається навколо гарячої точки. Вона вказує напрямок, яке в даний момент відповідає зображенню на 3D панорамі.

Спливаюче вікно є важливим елементом, який дозволяє зробити віртуальний тур більш інформативним. А розміщення інформації проходить без шкоди для перегляду самої панорами, так як після завантаження спливаючі вікна приховані від користувача і з'являються тільки після того, як ми його активуємо. Відкриття та закриття спливаючих вікон може супроводжуватися різними спец ефектами.

У віртуальний тур можна додавати відео. Виходить цікавий ефект. Його можна розмістити і всередині самої панорами, наприклад на зображенні екрану телевізора.

В якості звукового супроводу можуть використовуватися mp3 файли з описом місця зйомки або просто приємною мелодією. Можна закріпити свою власну мелодію для кожної панорами яка входить до складу віртуальної екскурсії і додати звук для кнопок або гарячих точок, який буде відтворюватися після натискання.

5.2 Програмне забезпечення для створення віртуальних турів

Biganto Visual - розробка для дизайнерів, архітекторів і візуалізаторів, що перетворює 3D модель інтер'єру або екстер'єру в тривимірний тур з вільним переміщенням і високою якістю візуалізації. Готовий тур можна відтворити в браузері на будь-якій платформі і в VR окулярах.

Krpano - програмний комплекс дозволяє об'єднувати панорами в віртуальні тури. Отриманий віртуальний тур можна розміщувати на сайті або демонструвати на комп'ютері.

Kolor Panotour Pro - візуальний редактор віртуальних турів, заснований на движку Krpano.

Pano2VR - візуальний редактор віртуальних турів, що використовує власний движок.

5.3 Реалізація віртуального туру

Після того, як було завершено редагування панорами, перейдемо до створення віртуального туру. Для цього будемо використовувати програму Pano2VR. Імпортуємо зображення панорами і виставляємо Тип: Циліндрична. Це обмежить можливість управління панорамою лише по горизонталі.

Потім, перейдемо до розділу Параметри проекції тут задаємо вигляд за замовчуванням для вихідної панорами, тобто кадр, з якого буде починатися перегляд віртуального туру. Для цього наведіть панораму туди, куди необхідно і натиснемо кнопку Зберегти в області Установки \ Обмеження. Також виставимо обмеження огляду, так як огляд вгору і вниз нам не потрібен. Інші параметри змінювати не будемо, натиснемо кнопку зберегти.

Далі переходимо до розділу Інформація про проект, цей розділ використовується для призначених для користувача даних, додавання інформації про зображення. Додана інформація тут же з'явиться в виведеному HTML шаблоні, якщо ви обрали автоматичного включення інформації. Також можна додати HTML-теги.

Проробляємо дану операцію з настройками для всіх панорам, з яких складається наш тур. Потім перейдемо до наступного розділу - Активні зони. У цьому розділі будуть налаштовуватися точки переходу між панорамами. Переходимо у вкладку Область активних зон і ставимо галочку Включити.

Створимо нову Активну зону. ID задається за замовчуванням - Активна зона 1. Для малювання точки переходу можна використовувати інструменти Олівець, Прямокутник, Еліпс і ін. Для зручності малювання також можна скористатися функцією Масштаб. Намалюємо область, яка буде інтерактивною, за допомогою інструменту Прямокутник. Активна зона буде червоного кольору. Активних зон може бути кілька. Для створення додаткових зон необхідно натиснути на кнопку поруч з ID поточної Активної зони у вигляді чистого аркуша.

Введемо ім'я гарячої точки в поле Тема. Це ім'я буде видно при наведенні курсору миші на активну область. Далі створимо зв'язок з іншою панорамою в турі. Для цього виберемо одне з посилань в області URL.

Повторимо ці дії для іншого приміщення і в результаті отримаємо готовий віртуальний тур, з можливістю переходу з однієї кімнати в іншу.

Далі перейдемо до додаванню кнопок і панелі управління. Для цього в розділі Експорт виберемо формат експорту Flash і натискаємо Додати, в результаті потрапляємо в меню налаштувань. У розділі оформлення зі списку виберемо потрібний скін. В даному випадку будемо використовувати controller_new.ggsk, також можна редагувати, запропонований варіант оформлення, або завантажити свій шаблон за допомогою команди Файл.

Відкриємо редактор оформлення. Тут можна додати свої кнопки, змінити стиль оформлення, задати звук при натисканні на елементи панелі управління, додати текст, додати спливаючі підказки або вставити інтерактивні елементи у вигляді анімацій. В даному випадком ми додамо дві кнопки переходу в різні приміщення.

Створимо панель управління з кнопками навігації, наближення і віддалення, переходу в повноекранний режим. Додамо два зображення, при натисканні на які буде здійснюватися переміщення з одного приміщення в інше.

Натискаємо два рази на зображення і переходимо до редагування його властивостей. На вкладці Actions / Modifiers створюємо дію, в результаті чого при натисканні на зображенні буде здійснюватися перехід до іншої панорамі.

Закінчимо редагування, натиснувши кнопку зберегти. Збережемо поточної шаблон під назвою controller_new_university.ggsk для його використання в іншому приміщенні. Повернемося до меню Налаштування експорту. Задамо ім'я експортованого файлу і натиснемо зберегти. З'явиться повідомлення з питанням підтвердження експорту.

За такою ж аналогією задаємо налаштування і для панорами іншої кімнати, змінивши лише адреси посилань на *.swf файли панорами.

Так як були створені Активні зони, переміщення може здійснюватися як з використанням панелі управління, так і безпосередньо при натисканні на арочний отвір. У підсумку, ми отримали готовий віртуальний тур, що

складається з декількох панорам, з панеллю керування, спливаючими вікнами з текстом і з можливістю переходу з одного приміщення в інше.

6 РОЗРОБКА ПРОГРАМНОЇ ЧАСТИНИ ПРОЕКТУ

6.1 Загальна архітектура системи

Базуючись на визначених вимогах для проекту, краще за все буде обрати модульну систему для реалізації мобільного додатку.

Мобільний додаток має мати у своїй складовій декілька незалежних один від одного модулів. Ізоляція модулів один від одного дозволить добитися кількох речей.

По-перше, це дозволить вільно використовувати код заново, не тягнучи за собою залежності одного функціонала від іншого. Це дозволить збільшити швидкість розробки програмного забезпечення і підвищить його якість. Зміни в одній частині не будуть тягти за собою зміни усієї програми.

По-друге, це дозволить легше тестувати написаний код. Модульність дозволить створювати заглушки для випадків, коли вхідними даними тесту буде результат виконання функціоналу, що розташований у іншому модулі. Це збільшить надійність роботи тестів та позбавить його залежностей від виконання необов’язкових для конкретних тестів частин коду.

По-третє, модульна структура дозволить використовувати незалежні один від одного модулі для створення інший мобільних додатків. У перспективі це дасть змогу для подальшого розвитку ідей, закладених у цьому проекті.

Отже, базуючись на визначених раніше вимогах проекту, визначимо необхідні модулі.

1. Модуль отримання даних з серверу CIST.
2. Модуль отримання даних з бази даних Firebase.
3. Модуль відображення розкладу викладача, групи або аудиторії.
4. Модуль відображення аудиторій, викладачів та груп для додавання у список.
5. Модуль відображення списку викладачів або аудиторій.

6. Модуль відображення конкретного викладача або аудиторії для отримання інформації.

7. Модуль сканування зображень та побудови доповненої реальності.

Наступна структура ізолює модуль сканування зображень та побудови доповненої реальності від модулів отримання та відображення інформації. Так як реалізація мобільного додатку на платформу iOS має на увазі використання мови програмування Swift або Objective-C, то реалізація взаємодії між ними буде використовувати протокольно-орієнтоване програмування, що виключить будь-які можливі залежності між модулями. Протоколи будуть описувати необхідні інтерфейси, що потрібно реалізувати розробнику конкретної програми. Модулі будуть працювати лише з відомим їм інтерфейсами, що виключить їх залежність від конкретної реалізації клієнтського додатку

6.2 Отримання інформації з серверу CIST.

Для зручності були реалізовані класи Endpoint, NetworkServices, Reachability, ResponseError та Parser. Розглянемо кожен з них.

Endpoint – містить в собі 3 поля даних.

HTTPMethod – використовується для опису методу запиту, існує багато різних методів, наприклад:

- GET – метод запитує уявлення ресурсу, запити з використанням цього методу можуть тільки отримувати дані;
- POST – використовується для відправки сутностей до певного ресурсу, часто викликає зміну стану або якісь побічні ефекти на сервері;
- PUT – замінює всі поточні уявлення ресурсу даними запиту.

Path – використовується для опису повного шляху запиту, наприклад, у додатку існує BaseURL – http://cist.nure.ua/ias/app/tt, поле Path зберігає в собі частину, яка вказує на конкретний запит:

- "/P_API_PODR_JSON" – шлях для запиту викладачів;
- "/P_API_GROUP_JSON" – шлях для запиту груп;

– "/P_API_AUDITORIES_JSON" – шлях для запиту аудиторій.

Parameters – поле яке необхідно для передачі параметрів на сервер, наприклад, якщо нам потрібен конкретний викладач, то ми повинні в параметрах передати його id, щоб не отримувати зайві дані.

Класс Reachability – містить в собі одне поле Bool.

Поле isInternetAvailable яке описує доступ до інтернету користувача. Використовується для відображення помилок, коли користувач без інтернету хоче отримати дані з серверу CIST, або з бази даних Firebase.

ResponseError – допоміжний клас.

Містить в собі опис помилок, для зручної обробки помилок. Існує 4 типи помилок:

– internetNotAvailable – користувач не має доступу до інтернету;

– dataMissed – з серверу не прийшло даних;

– invalidFormat – за серверу прийшли дані в невідомому форматі;

– failed – всі інші помилки.

NetworkServices – клас з описом запитів.

Містить в собі поле з BaseURL, а також методи запитів, JSON-request та String-request.

Лістинг 6.1 - Функція запиту з відповіддю у форматі String

```
func requestWithStringResponse(endpoint: Endpoint, completionHandler: ((Result<String>) -> Void)? = nil) {
    guard Reachability.isInternetAvailable else {
      completionHandler?(Result.failure(ResponseError.internetNotAvailable))
      return
    }


    Alamofire.request(baseURL + endpoint.path,
                      method: endpoint.method,
                      parameters: endpoint.parameters).responseString(queue: DispatchQueue.global()) {
                          networkResponse in
                       completionHandler?(networkResponse.result)
```

```
    }
  }
```

Функція, яка робить запит і отримує відгук від серверу у вигляді String, або якщо сталася помилка, то функція повертає помилку. Складається функція з початкової перевірки на інтернет, потім іде відправлення запиту на сервер, та відправка результатів з серверу у іншу функцію.

Лістинг 6.2 - Функція запиту з відповіддю у форматі JSON

```
func request(endpoint: Endpoint, completionHandler: ((Result<Any>) -> Void)? = nil) {
    guard Reachability.isInternetAvailable else {
      completionHandler?(Result.failure(ResponseError.internetNotAvailable))
      return
    }

    Alamofire.request(baseURL + endpoint.path,
                      method: endpoint.method,
                      parameters: endpoint.parameters).responseJSON(queue: DispatchQueue.global()) { networkResponse
in
                        completionHandler?(networkResponse.result)
    }
  }
```

Функція працює як і попередня, але повертає відгук від серверу у вигляді JSON.

Parser – містить 2 функції.

Лістинг 6.3 - Функція для перевірки і отримання даних

```
static func parseResponse(_ result: Result<Any>) -> Result<[String: JSON]> {
    guard result.isSuccess, let value = result.value else {
        return Result.failure(result.error ?? ResponseError.dataMissed)
    }

    guard let dictionary = JSON(value).dictionary,
        let status = dictionary["status"]?.string else {
            return Result.failure(ResponseError.dataMissed)
    }

    guard status == "OK" else {
        return Result.failure(ResponseError.failed(code: -1, description: "Some error"))
    }

    return Result.success(dictionary)
  }
```

Функція використовується для перевірки і отримання даних в необхідному для програми типі – [String: JSON].

Наступна функція працює, як і попередня, але для вихідних даних типу String.

Лістинг 6.4 - Функція для перевірки і отримання даних

```
static func parseStringResponse(_ result: Result<String>) -> Result<[String: JSON]> {
    guard result.isSuccess, let value = result.value else {
        return Result.failure(result.error ?? ResponseError.dataMissed)
    }

    let stringWithoutError = value.replacingOccurrences(of: "[\n}]", with: "[]")

    guard let dictionary = JSON(parseJSON: stringWithoutError).dictionary else {

        return Result.failure(ResponseError.dataMissed)
```

```
    }

    guard dictionary["error"] == nil else {
      return Result.failure(ResponseError.failed(code: -1, description: "Some error"))
    }

    return Result.success(dictionary)
  }
```

6.3 Отримання інформації з бази даних реального часу Firebase

Для бази даних реального часу був написаний модуль DatabaseManager, який містить в собі опис роботи з базою даних. Клас містить поля які описують необхідні сутності: rootDatabaseReference, teacherReference, auditoriesReference та методи отримання необхідних даних.

Лістинг 6.5 - Функція отримання даних з Firebase

```
func getFirebaseTeacherData(completion: @escaping ([String: Dictionary<String, Any>]?) -> ()) {
    self.teacherReference.observe(.value) { (snapshot) in
      completion(snapshot.value as? [String: Dictionary])
    }
```

Для реалізації методу використовуються методи з фрейморку Firebase, метод observe робить запит до бази даних та отримує необхідні дані, які потім використовуються у додатку.

Інші методи реалізовуються таким же чином, але з різними сутностями для отримання даних.

6.4 Відображення розкладу викладача, групи або аудиторії

Для початку був реалізований інтерфейс користувача, який охоплює всі необхідні для студенту дані. (рис 3.1). Також реалізоване меню в якому можна змінювати групу, аудиторію, викладача, для вибору необхідного, або для видалення добавленого розкладу (рис 3.2).

Інтерфейс забезпечує всією необхідною інформацією користувача, ви можете с цього екрану добавити до себе в список необхідного викладача, аудиторію, або групу, за допомогою допоміжного екрану, який викликається натисканням на кнопку + (рис 3.1) Також при натисканні на конкретну пару, ви зможете побачити її детальну інформацію. (рис 3.3). Інтерфейс був реалізовано для максимальної гнучкості, простоти і задоволення потребам користувачів.

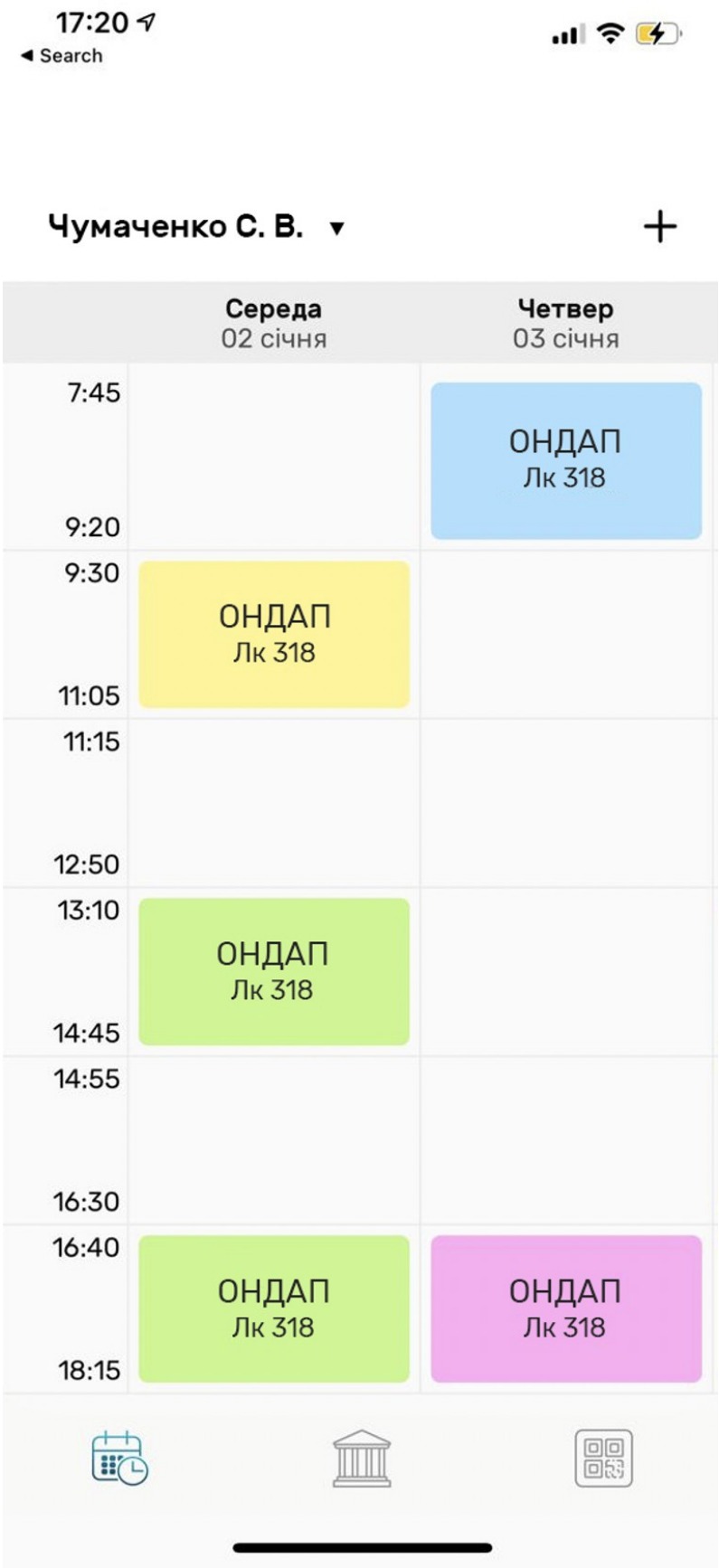

Рисунок 6.1 – Інтерфейс користувача для відображення розкладу

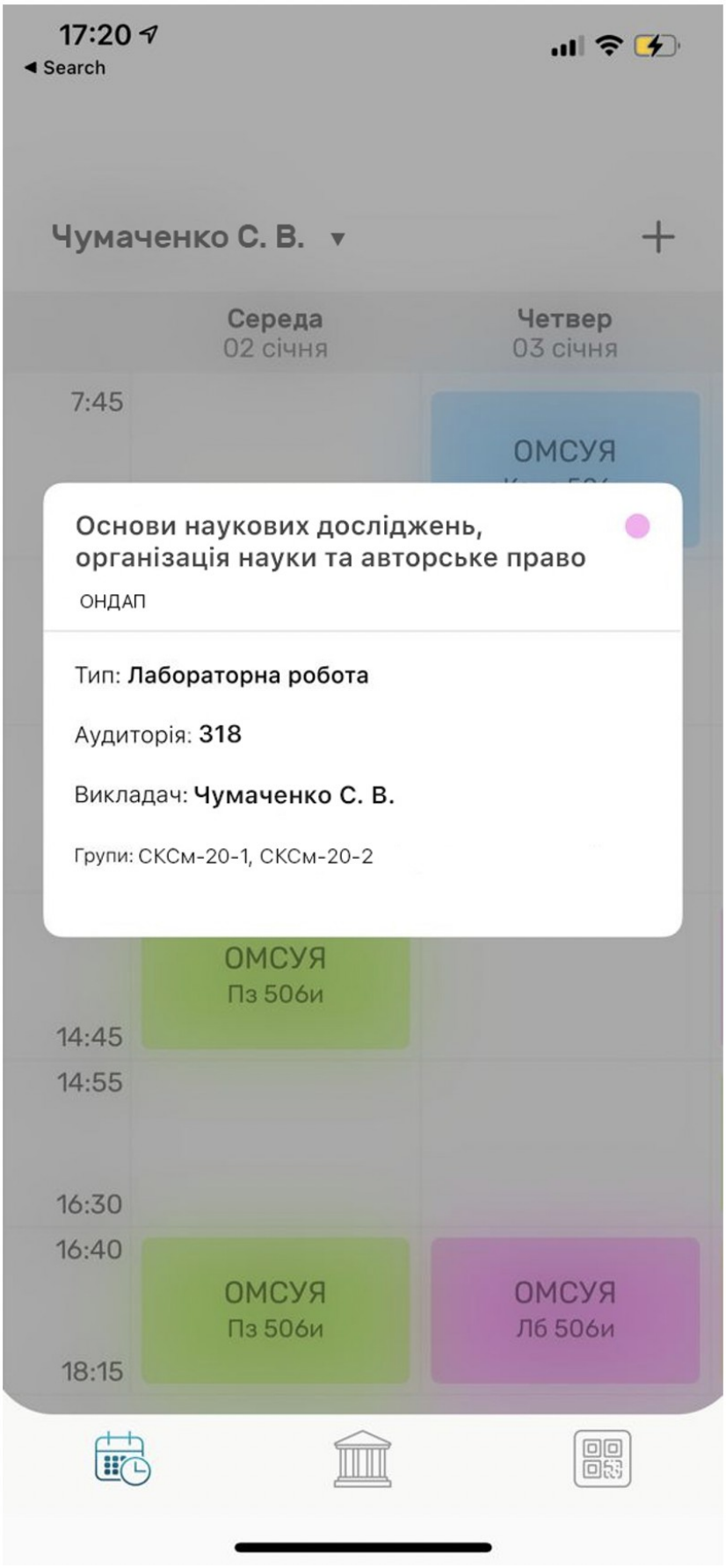

Рисунок 6.2 – Детальна інформація про вибрану пару

Тепер розглянемо реалізацію цих екранів.

Спочатку іде стандартна ініціалізація контролеру, а також налаштування графічного інтерфейсу в методі.

Лістинг 6.6 - Основна функція ініціалізації контролеру (фрагмент)

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

Налаштування включає в себе: присвоювання делегатів, установка протоколів, реєстрація комірок, а також налаштування кольорів.

Для відображення розкладу використовується бібліотека SpreadsheetView. Для відображення треба реалізувати протокол SpreadsheetViewDataSource до протоколу входять такі методи.

Лістинг 6.7 - Функція визначення кількості рядків

```
func numberOfRows(in spreadsheetView: SpreadsheetView) -> Int {
    return timeData.count
  }
```

Ця функція повертає кількість рядків у розкладі, timeData – масив у якому зберігаються дані о початку і кінці пари, наприклад, 7:45-9:20, 9:30-11:05, масив з цими даними масштабується в залежності з поточним обраним розкладом.

Лістинг 6.8 - Функція визначення кількості стовбців

```
func numberOfColumns(in spreadsheetView: SpreadsheetView) -> Int {
    return schedueleData.count
  }
```

Ця функція повертає кількість стовбців у розкладі, schedueleData – масив у якому зберігаються дані кожної пари.

Лістинг 6.9 - Функція визначення ширини комірки

```
func spreadsheetView(_ spreadsheetView: SpreadsheetView, widthForColumn column: Int) -> CGFloat {
    if column == 0 {
      return 65
    }
    return 150
  }
```

Ця функція визначає ширину комірки, як ми бачимо по дизайну, а також у реалізації, перший стовбець завжди меншої ширини, ніж стовбці у розкладі.

Лістинг 6.10 - Функція визначення висоти комірки

```
func spreadsheetView(_ spreadsheetView: SpreadsheetView, heightForRow row: Int) -> CGFloat {
    let countRows = CGFloat(timeData.count)
    let rowHeight = spreadsheetView.frame.height / countRows - 2 - (15 / countRows)
    if row == 0 {
      return rowHeight + 10
    }
    return rowHeight
  }
```

Ця функція визначає висоту комірки, як ми бачимо по дизайну, а також у реалізації, перший стовбець завжди меншої висоти, ніж стовбці у розкладі, а

також реалізована динамічна висота для комірки, в залежності від розміру екрану телефона.

Лістинг 6.11 - Функція визначення комірки

```
func spreadsheetView(_ spreadsheetView: SpreadsheetView, cellForItemAt indexPath: IndexPath) -> Cell? {
    if indexPath.column == 0 {
        let cell = spreadsheetView.dequeueReusableCell(withReuseIdentifier: String(describing: TimeCell.self), for: indexPath) as! TimeCell
        let time = timeData[indexPath.row]
        cell.setupCell(startTime: time.startTime.getStringTime(), endTime: time.endTime.getStringTime(), isFirstRow: indexPath.row == 0)
        cell.backgroundColor = UIColor(red: 0.98, green: 0.98, blue: 0.98, alpha: 1)
        return cell
    }

    let cell = spreadsheetView.dequeueReusableCell(withReuseIdentifier: String(describing: ClassCell.self), for: indexPath) as! ClassCell
    cell.setupCell(className: "ОМСУЯ", auditoryName: "лб 506и", classColor: testColors[Int.random(in: 0...3)], isFirstLine: indexPath.row == 0)
    cell.backgroundColor = UIColor(red: 0.98, green: 0.98, blue: 0.98, alpha: 1)
    return cell
  }
```

Ця функція повертає необхідну комірку, якщо це перший стовбець, то повертається TimeCell у якій відображується час пари, якщо ні, то повертається ClassCell, у якій відображується інформація о необхідному занятті. Також в функції іде налаштування кольору та текстів комірки.

Далі розглянемо реалізацію спливаючого меню для вибору, або видалення розкладу, для цього був створений допоміжний клас DropDownMenu, який реалізує протоколи UITableViewDataSource, UITableViewDelegate. Розглянемо детальніше методи цих протоколів.

Лістинг 6.12 - Функція визначення кількості рядків

```
func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    return data.count
}
```

Функція повертає кількість комірок у спливаючому меню, кількість залежить від кількості добавлених користувачем розкладів у додаток.

Лістинг 6.13 - Функція визначення комірки

```
func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    let cell = tableView.dequeueReusableCell(withIdentifier: "ShedueleNameTableViewCell", for: indexPath) as!
ShedueleNameTableViewCell
    cell.setupCell(name: data[indexPath.row])
    return cell
  }
```

Функція повертає необхідну комірку у спливаючому меню, а також налаштовує в ній необхідні дані.

Лістинг 6.14 - Функція видалення комірки.

```
func tableView(_ tableView: UITableView, trailingSwipeActionsConfigurationForRowAt indexPath: IndexPath) ->
UISwipeActionsConfiguration? {

    let deleteAction = UIContextualAction(style: .destructive, title: "Remove") { (action, sourceView,
completionHandler) in
      print("index path of delete: \(indexPath)")
      completionHandler(true)
    }
    deleteAction.image = UIImage(named: "deleteIcon")
    let swipeConfig = UISwipeActionsConfiguration(actions: [deleteAction])
    swipeConfig.performsFirstActionWithFullSwipe = true
    return swipeConfig
```

Ця функція реалізовує видалення комірки за допомогою свайпу по строчці, після свайпу з’являється кнопка видалення.

Для показу та приховування спливаючого меню, реалізовано натискання на кнопку.

Лістинг 6.15 - Функція показу та приховування спливаючого меню

```
@IBAction private func showDropDownButtonTouch(_ sender: UIButton) {
    dropDownHeightConstraint.constant = dropDownHeightConstraint.constant == 0 ? 141 : 0
    UIView.animate(withDuration: 0.5) {
      self.view.layoutIfNeeded()
    }
  }
```

В функції змінюється висота меню, а за допомогою стандартного методу UIView.animate зміна висоти відбувається анімовано.

6.5 Відображення аудиторій, викладачів та груп для додавання у список.

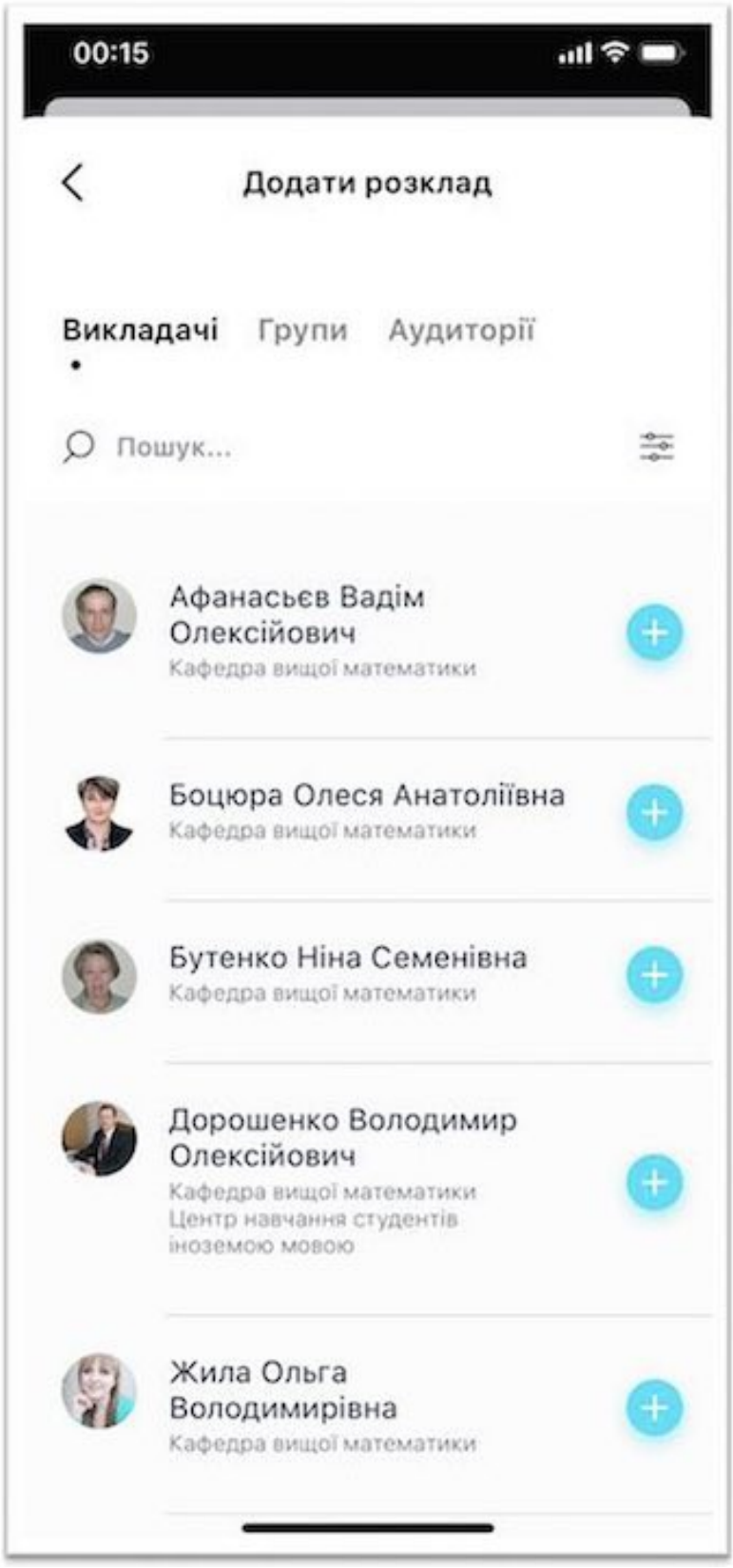

Рисунок 6.3 – Відображення аудиторій, викладачів та груп для додавання у список

На даному екрані реалізоване додавання викладачів, аудиторій, груп у

розклад. Для кожної із категорій реалізовано пошук, а також фільтри. Пошук викладачів відбувається за допомогою імені, пошук аудиторій, за допомогою номеру аудиторії та груп, за допомогою назви групи. Після того, як знайшов потрібного викладача, аудиторію, або групу, користувач повинен натиснути на кнопку +, та перейти назад до розкладу, де його чекає все перевантажена таблиця з розкладом.

Тепер розглянемо реалізацію цих екранів.

Спочатку іде стандартна ініціалізація контролеру, ініціалізація даних, а також налаштування графічного інтерфейсу в методі.

Лістинг 6.16 - Основна функція ініціалізації контролеру (фрагмент)

```
override func viewDidLoad() {
super.viewDidLoad()
setupView()
setupData()
{…}
}
```

Налаштування включає в себе: присвоювання делегатів, установка протоколів, реєстрація комірок, а також налаштування кольорів.

В контролері існує пошук, а також фільтр викладачів, аудиторій або груп. Спочатку розглянемо роботу пошуку. Для поля пошуку використовується стандартний UITextField.

Лістинг 6.17 - Функція пошуку серед груп, аудиторій, викладачів

```
@objc func textFieldDidChange(_ textField: UITextField) {

    let currentSearchText = textField.text?.lowercased()
    guard let searchText = currentSearchText else { return }
    if isTeachers {
        if searchText == "" {
            searchTeacherData = filterTeacherData
            reloadData(isFilterScrolling: false)
            return
        }
        var localArray = [(teacher: Teacher, departments: [TeacherDepartment])]()
        for teacherData in filterTeacherData {
            let fullName = teacherData.teacher.fullName?.lowercased()
            guard let name = fullName else {continue}
            let nameRus = name.replacingOccurrences(of: "i", with: "и")
            if name.contains(searchText) || nameRus.contains(searchText) {
                localArray.append(teacherData)
            }
        }
        searchTeacherData = localArray
    } else if isGroups {

        if searchText == "" {
            searchGroupData = filterGroupData
            reloadData(isFilterScrolling: false)
            return
        }
        var localArray = [(group: Group, direction: DirectionGroup)]()
        for groupData in filterGroupData {
            let fullName = groupData.group.name?.lowercased()

            guard let name = fullName else {continue}
            let nameRus = name.replacingOccurrences(of: "i", with: "и")
            if name.contains(searchText) || nameRus.contains(searchText) {
                localArray.append(groupData)
            }
        }

        searchGroupData = localArray
    } else {
```

```
            if searchText == "" {
                searchAuditoryData = filterAuditoryData
                reloadData(isFilterScrolling: false)
                return
            }
            var localArray = [(auditory: Auditory, building: AuditoryBuilding)]()
            for auditoryData in filterAuditoryData {
                let auditoryName = auditoryData.auditory.shortName?.lowercased()

                guard let name = auditoryName else {continue}
                let nameRus = name.replacingOccurrences(of: "i", with: "и")
                if name.contains(searchText) || nameRus.contains(searchText) {
                    localArray.append(auditoryData)
                }
            }
            searchAuditoryData = localArray
        }
        reloadData(isFilterScrolling: false)
    }
```

В функції реалізовуються 3 стану, перший – Викладачі, другий – Групи, третій – Аудиторії. В залежності від введеного тексту, відбувається пошук, спочатку всі букви перетворюються на маленькі, щоб від регістру літер не залежав пошук, потім зрівнюється текст пошуку з іменем групи, аудиторії або викладача, після пошуку відбувається перезавантаження таблиці для відображення нових даних приклад у лістингу 3.18.

Лістинг 6.18 - Функція перезавантаження таблиці

```
func reloadData(isFilterScrolling: Bool = false) {
    mainTableView.reloadData()
    if mainTableView.numberOfRows(inSection: 0) > 0 {
        mainTableView.scrollToRow(at: IndexPath.init(row: 0, section: 0), at: .top, animated: false)
    }
    filterCollectionView.reloadData()
    if isFilterScrolling {
        filterCollectionView.setContentOffset(.zero, animated: false)
    }
```

```
  }
```

Дана функція повинна викликатись завжди, коли змінюються дані у таблиці, або у фільтрах. Перевірка на кількість рядків необхідна щоб уникнути помилок, коли в таблиці немає рядків.

Розглянемо функції зміни інтерфейсу кнопок.

Лістинг 6.19 - Функція зміни інтерфейсу кнопки при її виборі

```
private func selectedButton(button: UIButton) {
    button.setTitleColor(UIColor(red: 0.09, green: 0.13, blue: 0.22, alpha: 1), for: .normal)
    button.isUserInteractionEnabled = false
    blackCircleView.frame.origin = CGPoint(x: button.frame.origin.x + 5, y: button.frame.maxY + 8)
    closeDepartmentsFilter()
  }
```

Функція змінює кольори кнопки, а також робить її не активною. Також зміщуються позиції елементів інтерфейсу. Фільтри закриваються, тому що в різних категоріях, різні фільтри.

Лістинг 6.20 - Функція зміни інтерфейсу кнопки при виборі іншої

```
private func unselectedButton(button: UIButton) {
    button.setTitleColor(UIColor(red: 0.09, green: 0.13, blue: 0.22, alpha: 0.5), for: .normal)
    button.isUserInteractionEnabled = true
  }
```

Функція змінює кольори кнопки, а також робить її активною.

Далі опишемо функції, які викликаються при натисканні на кнопки інтерфейсу користувача.

Лістинг 6.21 - Функція обробки натискання на кнопку Аудиторії

```
@IBAction private func auditoriesButtonTouch(_ sender: UIButton) {
    mainTableView.separatorInset.left = 25
    isTeachers = false
    isGroups = false
    selectedButton(button: auditoriesButton)
    unselectedButton(button: groupsButton)
    unselectedButton(button: teachersButton)
  }
```

В функції змінюється графічний інтерфейс таблиці, а також використовуються функції вибору кнопок, які описувались вище.

Лістинг 6.22 - Функція обробки натискання на кнопку Викладачі

```
@IBAction private func teachersButtonTouch(_ sender: UIButton) {
    mainTableView.separatorInset.left = 80
    isTeachers = true
    isGroups = false
    selectedButton(button: teachersButton)
    unselectedButton(button: groupsButton)
    unselectedButton(button: auditoriesButton)
  }
```

Лістинг 6.23 - Функція обробки натискання на кнопку Групи

```
@IBAction private func groupsButtonTouch(_ sender: UIButton) {
    mainTableView.separatorInset.left = 25
    isTeachers = false
    isGroups = true

    selectedButton(button: groupsButton)
    unselectedButton(button: teachersButton)
    unselectedButton(button: auditoriesButton)
  }
```

Лістинг 6.24 - Функція обробки натискання на кнопку Пошуку

```
@IBAction func searchButtonTouch(_ sender: Any) {

    searchTeacherData = filterTeacherData
    searchAuditoryData = filterAuditoryData
    searchGroupData = filterGroupData

    if isSearchActive {
        isSearchActive = false
        searchIcon.image = UIImage(named: "SearchIcon")
        searchTextField.text = ""
        searchTextField.resignFirstResponder()
    } else {
        isSearchActive = true
        searchIcon.image = UIImage(named: "CloseIcon")
        searchTextField.becomeFirstResponder()
    }

    reloadData()
}
```

В функції анулюються дані, а також змінюються іконки інтерфейсу і відбувається перезавантаження головної таблиці.

Лістинг 6.25 - Функція обробки натискання на кнопку Фільтри

```
@IBAction func filterButtonTouch(_ sender: Any) {

    if mainTableViewTopConstraint.constant == 30 {
        mainTableViewTopConstraint.constant = 85

    } else {
        mainTableViewTopConstraint.constant = 30
    }

    UIView.animate(withDuration: 0.3, animations: {
        self.view.layoutIfNeeded()
        self.closeDepartmentsFilter()
    })
}
```

В функції відбувається анімований показ та приховування колекції з фільтрами, анімація реалізована за допомогою методу UIView.animate

Лістинг 6.26 - Функція обробки натискання на кнопку Назад

```
@IBAction func backButtonTouch(_ sender: Any) {
    dismiss(animated: true, completion: nil)
```

Функція яка повертає користувача на екран з розкладом.

Лістинг 6.27 - Функція закриття фільтрів

```
@objc func closeDepartmentsFilter() {
    selectedFacultyIndex = nil
    selectedFilterItem = nil
    selectedHousingIndex = nil
    selectedFloorIndex = nil
    selectedGroupFacultyIndex = nil
    selectedGroupFilterItem = nil
    filterTeacherData = teacherData
    filterAuditoryData = auditoryData
    filterGroupData = groupData
    reloadData(isFilterScrolling: true)
  }
```

В функції анулюються дані, які були вибрані в фільтрах, також оновлюється головна таблиця.

Для відображення списку викладачів, аудиторій, груп використовується стандартна бібліотека UIKit. Клас для відображення UITableView, для якого треба реалізувати протоколи UITableViewDataSource та UITableViewDelegate до протоколів входять такі методи.

Лістинг 6.28 - Функція кількості рядків для головної таблиці

```
func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    if isTeachers {
        return isSearchActive ? searchTeacherData.count : filterTeacherData.count
    } else if isGroups {
        return isSearchActive ? searchGroupData.count : filterGroupData.count
    } else {
        return isSearchActive ? searchAuditoryData.count : filterAuditoryData.count
    }}
```

Функція яка вертає кількість рядків в залежності з вибраною категорією, а також в залежності з активованим пошуком.

Лістинг 6.29 - Функція визначення комірки

```
func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {

    if isTeachers {
        let cell = tableView.dequeueReusableCell(withIdentifier: "TeacherTableViewCell", for: indexPath) as!
TeacherTableViewCell

        let teacherInfo = isSearchActive ? searchTeacherData[indexPath.row] : filterTeacherData[indexPath.row]

        var descriptionString = ""

        for department in teacherInfo.departments {

            if let name = department.fullName {
                descriptionString += name + "\n"
            }
        }

        descriptionString.removeLast()

        let name = (teacherInfo.teacher.fullName ?? "").trimmingCharacters(in: .whitespaces)

        cell.setupCell(name: name, description: descriptionString, image: teacherInfo.teacher.image)
```

```
            return cell
        } else if isGroups {
            let cell = tableView.dequeueReusableCell(withIdentifier: "GroupTableViewCell", for: indexPath) as!
GroupTableViewCell

            let groupInfo = isSearchActive ? searchGroupData[indexPath.row] : filterGroupData[indexPath.row]

            cell.setupCell(name: groupInfo.group.name ?? "", description: groupInfo.direction.shortName ?? "")
            return cell
        } else {
            let cell = tableView.dequeueReusableCell(withIdentifier: "AuditoryTableViewCell", for: indexPath) as!
AuditoryTableViewCell

            let auditoryInfo = isSearchActive ? searchAuditoryData[indexPath.row] : filterAuditoryData[indexPath.row]

            var nameText = auditoryInfo.auditory.shortName ?? ""
            var housingName = (auditoryInfo.building.shortName ?? "").uppercased()
            var floor = auditoryInfo.auditory.floor ?? "0"

            if floor.isEmpty {floor = "0"}

            switch housingName {
            case "I": housingName = "корпус И"
            case "З": housingName = "корпус З"
            default: housingName = "главный корпус"
            }

            nameText = nameText.replacingOccurrences(of: "_", with: "")


            cell.setupCell(name: nameText, floor: floor, housing: housingName)

            return cell
        }
    }
```

Ця функція повертає необхідну комірку, якщо вибрана вкладка з викладачами, то повертається TeacherTableViewCell у якій відображується фотографія та кафедра викладача, якщо вибрана вкладка з аудиторіями, або групами, то повертається GroupTableViewCell, або AuditoryTableViewCell, у

якій відображується назва та кафедра, або поверх аудиторії. Також в функції іде налаштування інтерфейсів комірки.

Далі розглянемо реалізацію фільтрів для реалізації використовується стандартна бібліотека UIKit. Клас для відображення UICollectionView, для якого треба реалізувати протоколи UICollectionViewDataSource та UICollectionViewDelegate до протоколів входять такі методи.

Лістинг 6.30 - Функція визначення кількість фільтрів

```
func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection section: Int) -> Int {
    if isTeachers {
        guard let filterDataIndex = selectedFacultyIndex else { return filterData.count }
        return filterData[filterDataIndex].departments.count
    } else if isGroups {
        guard let filterDataIndex = selectedGroupFacultyIndex else {return groupsFilterData.count }
        return groupsFilterData[filterDataIndex].directions.count
    } else {
        guard let filterDataIndex = selectedHousingIndex else {return auditoryFilterData.count }
        return auditoryFilterData[filterDataIndex].floors.count
    }
}
```

Функція повертає кількість фільтрів при різних умовах, в залежності від вибраної вкладки, а також в залежності від вибраного раніше фільтру, так як фільтри мають розкриття, після першого вибору.

Лістинг 6.31 - Функція визначення комірки

```
func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath) ->
UICollectionViewCell {
    let cell = collectionView.dequeueReusableCell(withReuseIdentifier: "FilterCollectionViewCell", for: indexPath)
as! FilterCollectionViewCell
```

```
    if isTeachers {
        var shortName: String?

        if let filterDataIndex = selectedFacultyIndex {
            shortName = filterData[filterDataIndex].departments[indexPath.row].shortName
        } else {
            shortName = filterData[indexPath.row].shortName
        }

        cell.setupCell(title: (shortName ?? "").uppercased(), isSelected: indexPath == selectedFilterItem)
    } else if isGroups {
        var shortName: String?

        if let filterDataIndex = selectedGroupFacultyIndex {
            shortName = groupsFilterData[filterDataIndex].directions[indexPath.row].fullName
        } else {
            shortName = groupsFilterData[indexPath.row].fullName
        }

        cell.setupCell(title: (shortName ?? "").uppercased(), isSelected: indexPath == selectedGroupFilterItem)

    } else {
        var shortName: String?

        if let filterDataIndex = selectedHousingIndex {
            shortName = "\(auditoryFilterData[filterDataIndex].floors[indexPath.row]) этаж"
        } else {



            shortName = auditoryFilterData[indexPath.row].housingName
        }

        cell.setupCell(title: (shortName ?? "").uppercased(), isSelected: indexPath.row == selectedFloorIndex)
    }

    return cell
}
```

Ця функція повертає необхідну комірку. Також в функції іде налаштування інтерфейсів та даних комірки.

Лістинг 6.32 - Функція визначення розмірів комірки

```
func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout: UICollectionViewLayout,
sizeForItemAt indexPath: IndexPath) -> CGSize {

    let font = UIFont(name: "SFUIText-Semibold", size: 13)
    let fontAttributes = [NSAttributedString.Key.font: font]
    var shortName: String?

    if isTeachers {
      if let filterDataIndex = selectedFacultyIndex {
        shortName = filterData[filterDataIndex].departments[indexPath.row].shortName
      } else {
        shortName = filterData[indexPath.row].shortName
      }
    } else if isGroups {
      if let filterDataIndex = selectedGroupFacultyIndex {
        shortName = groupsFilterData[filterDataIndex].directions[indexPath.row].fullName
      } else {
        shortName = groupsFilterData[indexPath.row].fullName
      }
    } else {
      if let filterDataIndex = selectedHousingIndex {
        shortName = "\(auditoryFilterData[filterDataIndex].floors[indexPath.row]) этаж"
      } else {
        shortName = auditoryFilterData[indexPath.row].housingName
      }
    }

    let myText = (shortName ?? "").uppercased()
    let size = (myText as NSString).size(withAttributes: fontAttributes)

    return CGSize(width: size.width + 50, height: 25)
  }
```

Ця функція визначає розмір комірки фільтрів в залежності від даних, висота завжди статична, але ширина змінюється в залежності від необхідної строки.

Лістинг 6.33 - Функція визначення заголовку фільтрів

```
  func collectionView(_ collectionView: UICollectionView, viewForSupplementaryElementOfKind kind: String, at indexPath: IndexPath) -> UICollectionReusableView {
      switch kind {

      case UICollectionView.elementKindSectionHeader:
          let headerView = collectionView.dequeueReusableSupplementaryView(ofKind: kind, withReuseIdentifier: "Header", for: indexPath)
          let arrowImageView = UIImageView(image: UIImage(named: "ArrowImage"))
          arrowImageView.center = headerView.center
          arrowImageView.frame.origin.x = headerView.frame.width - arrowImageView.frame.width
          headerView.addSubview(arrowImageView)

          let tapGestureRecognizer = UITapGestureRecognizer(target:self, action:#selector(closeDepartmentsFilter))
          headerView.addGestureRecognizer(tapGestureRecognizer)

          return headerView
      default: assert(false, "Unexpected element kind")
      }
  }
```

В функції відбувається налаштування заголовку фільтрів, заголовок фільтрів представляє з себе кнопку назад, коли перша категорія фільтрів відрита, якщо перша категорія фільтрів закрита, то кнопки назад не буде, а ширина заголовку буде дорівнювати 0.

Лістинг 6.34 - Функція вибору фільтрів (фрагмент)

```
func collectionView(_ collectionView: UICollectionView, didSelectItemAt indexPath: IndexPath) {…}
```

В функції є 3 стану, в залежності від категорії, також фільтри залежать від того, відкритий, чи закритий перший рівень фільтрів. В даній функції сортируються дані в залежності від вибраних фільтрів, самі фільтри беруться з заздалегідь підготовлених даних, які були отримані з серверу CIST. У кожного

викладача, аудиторії та групи, є свої фільтри к яким вони відносяться і за рахунок цього і відбувається сортування даних за фільтрами.

6.6 Відображення списку викладачів або аудиторій

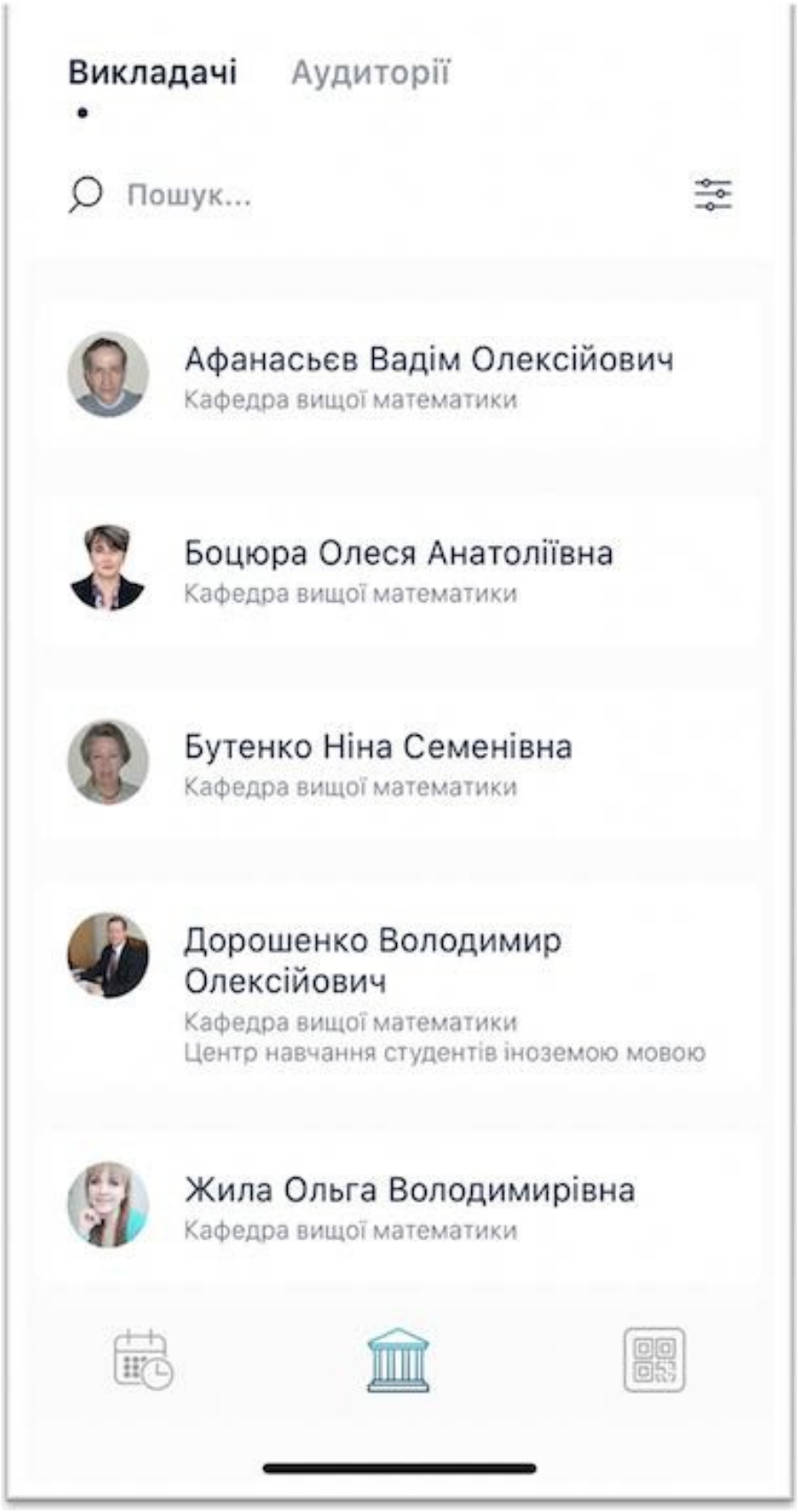

Рисунок 6.4 – Відображення списку викладачів або аудиторій

Тепер розглянемо реалізацію цього екрану.

Спочатку іде стандартна ініціалізація контролеру, ініціалізація даних, а також налаштування графічного інтерфейсу в методі.

Лістинг 6.35 - Основна функція ініціалізації контролеру (фрагмент)

```
override func viewDidLoad() {
super.viewDidLoad()
setupView()
setupData()
{…}
}
```

Налаштування включає в себе: присвоювання делегатів, установка протоколів, реєстрація комірок, а також налаштування кольорів.

В контролері існує пошук, а також фільтр викладачів, аудиторій або груп. Спочатку розглянемо роботу пошуку. Для поля пошуку використовується стандартний UITextField.

Лістинг 6.36 - Функція пошуку серед аудиторій та викладачів

```
@objc func textFieldDidChange(_ textField: UITextField) {

    let currentSearchText = textField.text?.lowercased()

    guard let searchText = currentSearchText else { return }

    var isNotFound = false

    if isTeachers {

       if searchText == "" {
          searchTeacherData = filterTeacherData
```

```
            reloadData(isFilterScrolling: false)
            return
        }

        var localArray = [(teacher: Teacher, departments: [TeacherDepartment])]()

        for teacherData in filterTeacherData {
            let fullName = teacherData.teacher.fullName?.lowercased()

            guard let name = fullName else {continue}
            let nameRus = name.replacingOccurrences(of: "i", with: "и")
            if name.contains(searchText) || nameRus.contains(searchText) {
                localArray.append(teacherData)
            }
        }

        isNotFound = localArray.count != 0

        searchTeacherData = localArray
    } else {
        if searchText == "" {
            searchAuditoryData = filterAuditoryData
            reloadData(isFilterScrolling: false)
            return
        }

        var localArray = [(auditory: Auditory, building: AuditoryBuilding)]()

        for auditoryData in filterAuditoryData {
            let auditoryName = auditoryData.auditory.shortName?.lowercased()

            guard let name = auditoryName else {continue}

            let nameRus = name.replacingOccurrences(of: "i", with: "и")
            if name.contains(searchText) || nameRus.contains(searchText) {
                localArray.append(auditoryData)
            }
        }

        isNotFound = localArray.count != 0

        searchAuditoryData = localArray
```

```
}
```

В функції реалізовуються 2 стану, перший – Викладачі, другий – Аудиторії. В залежності від введеного тексту, відбувається пошук, спочатку всі букви перетворюються на маленькі, щоб від регістру літер не залежав пошук, потім зрівнюється текст пошуку з іменем аудиторії або викладача, після пошуку відбувається перезавантаження таблиці для відображення нових даних приклад у лістингу 3.34. Також реалізований стан, коли пошук здійснити неможливо, тому що таких даних не має, користувач отримує інформацію о тому, що йому треба змінити строку пошуку.

Лістинг 6.37 - Функція перезавантаження таблиці

```
func reloadData(isFilterScrolling: Bool = false) {
    mainTableView.reloadData()
    if mainTableView.numberOfRows(inSection: 0) > 0 {
      mainTableView.scrollToRow(at: IndexPath.init(row: 0, section: 0), at: .top, animated: false)
    }
    filterCollectionView.reloadData()
    if isFilterScrolling {
      filterCollectionView.setContentOffset(.zero, animated: false)
```

Дана функція повинна викликатись завжди, коли змінюються дані у таблиці, або у фільтрах. Перевірка на кількість рядків необхідна щоб уникнути помилок, коли в таблиці немає рядків.

Лістинг 6.38 - Функція зміни інтерфейсу кнопки при її виборі

```
private func selectedButton(button: UIButton) {
    button.setTitleColor(UIColor(red: 0.09, green: 0.13, blue: 0.22, alpha: 1), for: .normal)
    button.isUserInteractionEnabled = false
    blackCircleView.frame.origin = CGPoint(x: button.frame.origin.x + 5, y: button.frame.maxY + 8)
    closeDepartmentsFilter()
  }
```

Функція змінює кольори кнопки, а також робить її не активною. Також зміщуються позиції елементів інтерфейсу. Фільтри закриваються, тому що в різних категоріях, різні фільтри.

Лістинг 6.39 - Функція зміни інтерфейсу кнопки при виборі іншої

```
private func unselectedButton(button: UIButton) {
    button.setTitleColor(UIColor(red: 0.09, green: 0.13, blue: 0.22, alpha: 0.5), for: .normal)
    button.isUserInteractionEnabled = true
  }
```

Функція змінює кольори кнопки, а також робить її активною.

Далі опишемо функції, які викликаються при натисканні на кнопки інтерфейсу користувача.

Лістинг 6.40 - Функція обробки натискання на кнопку Аудиторії

```
@IBAction private func auditoriesButtonTouch(_ sender: UIButton) {
    isTeachers = false
    selectedButton(button: auditoriesButton)
    unselectedButton(button: teachersButton)
  }
```

В функції використовуються функції вибору кнопок, які описувались вище.

Лістинг 6.41 - Функція обробки натискання на кнопку Викладачі

```
@IBAction private func teachersButtonTouch(_ sender: UIButton) {
    isTeachers = true
    selectedButton(button: teachersButton)
    unselectedButton(button: auditoriesButton)
  }
```

Лістинг 6.42 - Функція обробки натискання на кнопку Пошуку

```
@IBAction func searchButtonTouch(_ sender: Any) {

    searchTeacherData = filterTeacherData
    searchAuditoryData = filterAuditoryData

    notFoundLabel.isHidden = true
    notFoundImageView.isHidden = true

    if isSearchActive {
      isSearchActive = false
      closeImageView.isHidden = true
      searchTextField.text = ""
      searchTextField.resignFirstResponder()
    } else {
      isSearchActive = true
      closeImageView.isHidden = false
      searchTextField.becomeFirstResponder()
    }

    reloadData()
  }
```

В функції анулюються дані, а також змінюються іконки інтерфейсу і відбувається перезавантаження головної таблиці.

Лістинг 6.43 - Функція обробки натискання на кнопку Фільтри

```
@IBAction func filterButtonTouch(_ sender: Any) {

    if mainTableViewTopConstraint.constant == 20 {
      mainTableViewTopConstraint.constant = 80
    } else {
      mainTableViewTopConstraint.constant = 20
    }

    UIView.animate(withDuration: 0.3, animations: {
      self.view.layoutIfNeeded()
      self.closeDepartmentsFilter()
    })
  }
```

В функції відбувається анімований показ та приховування колекції з фільтрами, анімація реалізована за допомогою методу UIView.animate.

Функція яка повертає користувача на екран з розкладом.

Лістинг 6.44 - Функція закриття фільтрів

```
@objc func closeDepartmentsFilter() {
    selectedFacultyIndex = nil
    selectedFilterItem = nil
    selectedHousingIndex = nil
    selectedFloorIndex = nil
    filterTeacherData = teacherData
    filterAuditoryData = auditoryData
    reloadData(isFilterScrolling: true)
  }
```

В функції анулюються дані, які були вибрані в фільтрах, також оновлюється головна таблиця.

Для відображення списку викладачів, аудиторій, груп використовується стандартна бібліотека UIKit. Клас для відображення UITableView, для якого треба реалізувати протоколи UITableViewDataSource та UITableViewDelegate до протоколів входять такі методи.

Лістинг 6.45 - Функція кількості рядків для головної таблиці

```
func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    if isTeachers {
        return isSearchActive ? searchTeacherData.count : filterTeacherData.count
    } else {
        return isSearchActive ? searchAuditoryData.count : filterAuditoryData.count
    }
}
```

Функція яка вертає кількість рядків в залежності з вибраною категорією, а також в залежності з активованим пошуком.

Лістинг 6.46 - Функція визначення комірки

```
func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {

    if isTeachers {
        let cell = tableView.dequeueReusableCell(withIdentifier: "TeacherTableViewCell", for: indexPath) as!
TeacherTableViewCell

        let teacherInfo = isSearchActive ? searchTeacherData[indexPath.row] : filterTeacherData[indexPath.row]

        var descriptionString = ""

        for department in teacherInfo.departments {
            if let name = department.fullName {
                descriptionString += name + "\n"
            }
        }

        descriptionString.removeLast()

        let name = (teacherInfo.teacher.fullName ?? "").trimmingCharacters(in: .whitespaces)

        cell.setupCell(name: name, description: descriptionString, image: teacherInfo.teacher.image)

        return cell
```

```
    } else {
        let cell = tableView.dequeueReusableCell(withIdentifier: "AuditoryTableViewCell", for: indexPath) as!
AuditoryTableViewCell

        let auditoryInfo = isSearchActive ? searchAuditoryData[indexPath.row] : filterAuditoryData[indexPath.row]

        var nameText = auditoryInfo.auditory.shortName ?? ""
        var housingName = (auditoryInfo.building.shortName ?? "").uppercased()
        var floor = auditoryInfo.auditory.floor ?? "0"

        if floor.isEmpty {floor = "0"}

        switch housingName {
           case "I": housingName = "корпус I"
           case "3": housingName = "корпус 3"
           default: housingName = "головний корпус"
        }

        nameText = nameText.replacingOccurrences(of: "_", with: "")

        cell.setupCell(name: nameText, floor: floor, housing: housingName)

        return cell
    }
}
```

Ця функція повертає необхідну комірку, якщо вибрана вкладка з викладачами, то повертається TeacherTableViewCell у якій відображується фотографія та кафедра викладача, якщо вибрана вкладка з аудиторіями, то повертається AuditoryTableViewCell, у якій відображується назва та кафедра. Також в функції іде налаштування інтерфейсів комірки.

Далі розглянемо реалізацію фільтрів для реалізації використовується стандартна бібліотека UIKit. Клас для відображення UICollectionView, для якого треба реалізувати протоколи UICollectionViewDataSource та

UICollectionViewDelegate до протоколів входять такі методи.

Лістинг 6.47 - Функція визначення кількість фільтрів

```
func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection section: Int) -> Int {
    if isTeachers {
        guard let filterDataIndex = selectedFacultyIndex else { return filterData.count }
        return filterData[filterDataIndex].departments.count
    } else {
        guard let filterDataIndex = selectedHousingIndex else {return auditoryFilterData.count }
        return auditoryFilterData[filterDataIndex].floors.count
    }
}
```

Функція повертає кількість фільтрів при різних умовах, в залежності від вибраної вкладки, а також в залежності від вибраного раніше фільтру, так як фільтри мають розкриття, після першого вибору.

Лістинг 6.48 - Функція визначення комірки

```
func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath) ->
UICollectionViewCell {
    let cell = collectionView.dequeueReusableCell(withReuseIdentifier: "FilterCollectionViewCell", for: indexPath)
as! FilterCollectionViewCell

    if isTeachers {
        var shortName: String?

        if let filterDataIndex = selectedFacultyIndex {
            shortName = filterData[filterDataIndex].departments[indexPath.row].shortName
        } else {

            shortName = filterData[indexPath.row].shortName
        }

        cell.setupCell(title: (shortName ?? "").uppercased(), isSelected: indexPath == selectedFilterItem)
    } else {
        var shortName: String?
```

```
            if let filterDataIndex = selectedHousingIndex {
                shortName = "\(auditoryFilterData[filterDataIndex].floors[indexPath.row]) этаж"
            } else {
                shortName = auditoryFilterData[indexPath.row].housingName
            }

            cell.setupCell(title: (shortName ?? "").uppercased(), isSelected: indexPath.row == selectedFloorIndex)
        }


        return cell
    }
```

Ця функція повертає необхідну комірку. Також в функції іде налаштування інтерфейсів та даних комірки.

Лістинг 6.49 - Функція визначення розмірів комірки

```
func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout: UICollectionViewLayout,
sizeForItemAt indexPath: IndexPath) -> CGSize {

    let font = UIFont(name: "SFUIText-Semibold", size: 13)
    let fontAttributes = [NSAttributedString.Key.font: font]
    var shortName: String?

    if isTeachers {
        if let filterDataIndex = selectedFacultyIndex {
            shortName = filterData[filterDataIndex].departments[indexPath.row].shortName
        } else {
            shortName = filterData[indexPath.row].shortName
        }
    } else {
        if let filterDataIndex = selectedHousingIndex {
            shortName = "\(auditoryFilterData[filterDataIndex].floors[indexPath.row]) этаж"
        } else {
            shortName = auditoryFilterData[indexPath.row].housingName
        }
    }
```

```
        let myText = (shortName ?? "").uppercased()
        let size = (myText as NSString).size(withAttributes: fontAttributes)

        return CGSize(width: size.width + 50, height: 25)
    }
```

Ця функція визначає розмір комірки фільтрів в залежності від даних, висота завжди статична, але ширина змінюється в залежності від необхідної строки.

Лістинг 6.50 - Функція визначення заголовку фільтрів

```
    func collectionView(_ collectionView: UICollectionView, viewForSupplementaryElementOfKind kind: String, at indexPath: IndexPath) -> UICollectionReusableView {
        switch kind {

        case UICollectionView.elementKindSectionHeader:
            let headerView = collectionView.dequeueReusableSupplementaryView(ofKind: kind, withReuseIdentifier: "Header", for: indexPath)
            let arrowImageView = UIImageView(image: UIImage(named: "ArrowImage"))
            arrowImageView.center = headerView.center
            arrowImageView.frame.origin.x = headerView.frame.width - arrowImageView.frame.width
            headerView.addSubview(arrowImageView)

            let tapGestureRecognizer = UITapGestureRecognizer(target:self, action:#selector(closeDepartmentsFilter))
            headerView.addGestureRecognizer(tapGestureRecognizer)

            return headerView
        default: assert(false, "Unexpected element kind")
```

В функції відбувається налаштування заголовку фільтрів, заголовок фільтрів представляє з себе кнопку назад, коли перша категорія фільтрів відрита, якщо перша категорія фільтрів закрита, то кнопки назад не буде, а ширина заголовку буде дорівнювати 0.

6.7 Відображення конкретного викладача або аудиторії для отримання інформації

Цей екран визивається користувачем при натисканні на головному екрані вибору викладача, аудиторії. Після натискання відкривається екран де знаходиться вся інформація о викладачах та аудиторіях. Ми бачимо інформацію починаючи з кар’єри викладача, закінчуючи його електронним адресом (рис. 3.6). Також з цього екрану можна перейти на розклад викладача, або одразу добавити його собі в розклад.

Рисунок 6.5 – Відображення інформації про конкретного викладача

Тепер розглянемо реалізацію цього екрану.

Спочатку іде стандартна ініціалізація контролеру, ініціалізація даних, а також налаштування графічного інтерфейсу в методі.

Лістинг 6.51 - Основна функція ініціалізації контролеру (фрагмент)

```
override func viewDidLoad() {
super.viewDidLoad()
setupView()
setupData()
{…}
}
```

Налаштування включає в себе: присвоювання делегатів, установка протоколів, реєстрація комірок, а також налаштування кольорів.

Контролер не несе в собі тяжкої логічної частини, тому що використовується лише для відображення інформації, яку отримує з попереднього екрану. Всі дані для відображення заповнюються у функції setupView та setupData, які викликаються у методі viewDidLoad.

6.8 Сканування зображень та побудова доповненої реальності

В даному модулі відбувається налаштування сцени доповненої реальності, сканування карток викладачів та аудиторій, потім подальше відображення інформації в 3D, а також можливості переходу на екран з повною інформацією на екрані смартфону, який ми розглядали раніше.

Розглянемо логіку налаштування доповненої реальності.

Спочатку іде стандартна ініціалізація контролеру, ініціалізація даних, а також налаштування графічного інтерфейсу в методі.

Лістинг 6.52 - Основна функція ініціалізації контролеру (фрагмент)

```
override func viewDidLoad() {
super.viewDidLoad()
setupView()
{…}
}
```

Налаштування включає в себе: присвоювання делегатів, установка протоколів, реєстрація комірок, а також налаштування кольорів.

Далі викликається метод запуску сесії доповненої реальності.

Лістинг 6.53 - Запуск сесії доповненої реальності

```
func runSession() {
      // Load reference images to look for from "AR Resources" folder
      guard let referenceImages = ARReferenceImage.referenceImages(inGroupNamed: "AR Resources", bundle: nil)
else {
         fatalError("Missing expected asset catalog resources.")
      }

      // Create a session configuration
      let configuration = ARWorldTrackingConfiguration()

      // Add previously loaded images to ARScene configuration as detectionImages
      configuration.detectionImages = referenceImages

      // Run the view's session
      sceneView.session.run(configuration)
   }
```

У даному методі іде налаштування доповненої реальності. В даному коді ми кажемо телефону, щоб він сканував всі зображення які у нього є в папці ARResources – це папка яка містить картки викладачів.

Далі стандартними методами відбувається пошук картки викладача, або аудиторії. Після знаходження картки викликається метод.

Лістинг 6.54 - Знаходження 2D картки в реальному світі

```
func renderer(_ renderer: SCNSceneRenderer, didAdd node: SCNNode, for anchor: ARAnchor) {
    guard let imageAnchor = anchor as? ARImageAnchor, let currentTeacherID = Int(imageAnchor.name!),
    let teacherInfo = DataManager.instance.getTeacher(with: currentTeacherID) else {
        return
    }

    currentTeacherInfo = teacherInfo.teacher

    if let imageString = teacherInfo.teacher.image, let name = teacherInfo.teacher.fullName {
        DispatchQueue.main.async {
            self.teacherView.isHidden = false
            self.tutorialView.isHidden = true
            self.teacherImage.sd_setImage(with: URL(string: imageString), completed: nil)
            self.teacherName.text = name

            var descriptionString = ""

            for department in teacherInfo.departments {
                if let name = department.fullName {
                    descriptionString += name + "\n"
                }
            }

            descriptionString.removeLast()

            self.teacherStatus.text = descriptionString
        }
    }

    if let removeAnchro = oldAnchor {
        sceneView.session.remove(anchor: removeAnchro)
    }
    imageNode?.removeFromParentNode()


    planeNode?.removeFromParentNode()

    // 1. Load plane's scene.
    let planeScene = SCNScene(named: "Models.scnassets/\(imageAnchor.name!).scn")!
    let planeNode = planeScene.rootNode.childNode(withName: "plane", recursively: true)!
```

```
    // 2. Calculate size based on planeNode's bounding box.
    let (min, max) = planeNode.boundingBox
    let size = SCNVector3Make(max.x - min.x, max.y - min.y, max.z - min.z)

    // 3. Calculate the ratio of difference between real image and object size.
    // Ignore Y axis because it will be pointed out of the image.
    let widthRatio = Float(imageAnchor.referenceImage.physicalSize.width)/size.x
    let heightRatio = Float(imageAnchor.referenceImage.physicalSize.height)/size.z
    // Pick smallest value to be sure that object fits into the image.
    let finalRatio = [widthRatio, heightRatio].min()! + 0.05

    // 4. Set transform from imageAnchor data.
    planeNode.transform = SCNMatrix4(imageAnchor.transform)

    // 5. Animate appearance by scaling model from 0 to previously calculated value.
    let appearanceAction = SCNAction.scale(to: CGFloat(finalRatio), duration: 3)
    appearanceAction.timingMode = .easeOut
    // Set initial scale to 0.
    planeNode.scale = SCNVector3Make(0.001, 0.001, 0.001)
    // Add to root node.
    sceneView.scene.rootNode.addChildNode(planeNode)
    // Run the appearance animation.
    planeNode.runAction(appearanceAction)

    self.oldAnchor = anchor
    self.planeNode = planeNode
    self.imageNode = node
}
```

У даному коді після знаходження необхідного якорю відбувається налаштування 3D сцени з інформацією викладача, або аудиторії. Спочатку створюється об’єкт з інформацією, потім його позиція, та кути нахилу присвоюються реальним координатам та кутам, які були отримані з сканування реальної 2D карточки.

Далі для підтримки правильної позиції в методі делегату сесії доповненої реальності викликається оновлення позиції і кутів нахилу.

Лістинг 6.55 - Оновлення позиції об’єкту

```
    func renderer(_ renderer: SCNSceneRenderer, updateAtTime time: TimeInterval) {
        guard let imageNode = imageNode, let planeNode = planeNode else {
            return
        }

        // 1. Unwrap animationInfo. Calculate animationInfo if it is nil.
        guard let animationInfo = animationInfo else {
            refreshAnimationVariables(startTime: time,
                                      initialPosition: planeNode.simdWorldPosition,
                                      finalPosition: imageNode.simdWorldPosition,
                                      initialOrientation: planeNode.simdWorldOrientation,
                                      finalOrientation: imageNode.simdWorldOrientation)
            return
        }

        // 2. Calculate new animationInfo if image position or orientation changed.
        if !simd_equal(animationInfo.finalModelPosition, imageNode.simdWorldPosition) ||
animationInfo.finalModelOrientation != imageNode.simdWorldOrientation {

            refreshAnimationVariables(startTime: time,
                                      initialPosition: planeNode.simdWorldPosition,
                                      finalPosition: imageNode.simdWorldPosition,
                                      initialOrientation: planeNode.simdWorldOrientation,
                                      finalOrientation: imageNode.simdWorldOrientation)
        }

        // 3. Calculate interpolation based on passedTime/totalTime ratio.
        let passedTime = time - animationInfo.startTime
        var t = min(Float(passedTime/animationInfo.duration), 1)
        // Applying curve function to time parameter to achieve "ease out" timing



        t = sin(t * .pi * 0.5)

        // 4. Calculate and set new model position and orientation.
        let f3t = simd_make_float3(t, t, t)
        planeNode.simdWorldPosition = simd_mix(animationInfo.initialModelPosition,
animationInfo.finalModelPosition, f3t)
        planeNode.simdWorldOrientation = simd_slerp(animationInfo.initialModelOrientation,
animationInfo.finalModelOrientation, t)
        //planeNode.simdWorldOrientation = imageNode.simdWorldOrientation
    }
```

За допомогою методу refreshAnimationVariables зміна позицій і кутів відбувається анімовано. Кожен раз при оновленні сесії доповненої реальності викликається цей метод і оновлюється позиція нашого об’єкту, який залежить від позиції реального об’єкту в реальному світі.

6.9 Приклади реалізації віртуального туру

Даний приклад показує лише частину функціоналу, повну версію туру з переглядом 360 панорам і навігацією можна переглянути у додатку. У версії 2D документу не можливо охопити повний показ 360 панорам. До атестаційної роботи в перезентації представлено відео для перегляда повного функціонала, без необхідності завантаження додатку з App Store.

Проведемо тур до кафедри АПОТ. Спочатку ми знаходимось в головному коридорі третього поверху (рис. 6.6).

Рисунок 6.6 – Головний коридор третього поверху

Після цього ми переміщуємось до двері кафедри АПОТ (рис. 6.7)

Рис. 6.7 – Коридор до кафедри АПОТ

Далі ми заходимо на кафедру АПОТ і опиняємось у головному коридорі кафедри (рис. 6.8)

Рисунок 6.8 – Головний коридор кафедри АПОТ

Після цього ми можемо рухатись до наступного коридору в якому аудиторія 318 та аудиторія для викладачів, або в аудиторію 320. Спочатку зайдемо в аудиторію 320 (рис. 6.9)

Рисунок 6.9 – Аудиторія 320 кафедри АПОТ

Після цього ми вийдемо з аудиторії 320 і перейдемо в коридор (рис. 6.10) до аудиторії викладачів та аудиторії 318.

Рисунок 6.10 – Коридор до аудиторії 318 кафедри АПОТ

Останнім нашим пунктом буде аудиторія 318 (рис. 6.11)

Рисунок 6.11 – Аудиторія 318 кафедри АПОТ

ВИСНОВКИ

В атестаційній роботі були розглянуті новітні технології доповненої реальності, а також технології баз даних реального часу, також проаналізована та розглянута робота з HTTP-запитами.

Було розглянуто можливість створення нового сервісу кіберуніверситету, який буде допомагати абітурієнтам та їх батькам.

Після собору даних і аналізу, був створений додаток для студентів Харківського національного університету радіоелектроніки, який дозволяє швидко отримати доступ до розкладу викладача, своєї групи, або навіть аудиторії. Також студент зможе за допомогою пошуку, або фільтрів знайти необхідного викладача, аудиторію та побачити всю розгорнуту інформацію о ньому.

Було створено 6 сферичних панорам з кафедри АПОТ для реалізації віртуального туру по університету.

Було використано метод сканування зображень в доповненій реальності, це допомогло створити функцію, коли кожен студент зможе навести телефон на картку викладача та аудиторії і побачити всю інформацію в тривимірному світі.

Програмна реалізація алгоритму керування здійснена на мові програмування Swift у середовищі розробки XCode. Моделювання розробленого додатку проводилося інструментальними засобами XCode.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Мытников А.Н., Мытникова Е.А., Кузнецова Л.Н., Солин С.Ю. Технологии разработки мобильных приложений // Теория и практика современной науки. – 2016. – № 4 (10). – С. 504-507.

2. Мытников А.Н., Мытникова Е.А. История развития дополненной реальности [Электронный ресурс] / Журнал «novaum.ru» Выпуск 5. 27 Июня 2017 г. – Режим доступу: www / URL: http://novaum.ru/public/p148

3. Augmented Reality [Електронний ресурс] / Apple Developer – Режим доступу: www / URL: https://developer.apple.com/augmented-reality

4. Apple Developer Documentation [Електронний ресурс] / Apple Developer – Режим доступу: www / URL: https://developer.apple.com/documentation

5. Firebase Realtime Database [Електронний ресурс] / Google – Режим доступу: www / URL: https://firebase.google.com/docs/database

6. Хаханов В.И. Киберсоциальная система – умный кибер-университет / В. И. Хаханов, Е. И. Литвинова, С. В. Чумаченко, А. С. Мищенко. – Радіоелектронні і Комп'ютерні Системи.– 2016.– № 5 (79). – С. 187-194.

7. Створення віртуальних турів та панорам [Електронний ресурс] / Panorama – Режим доступу: www / URL: http://pano.su/

8. Віртуальні тури та панорами [Електронний ресурс] / 1Panorama – Режим доступу: www / URL: http://1panorama.ru/

9. Yanko A. D. Importance of practical use of delegates in development of mobile applications / Yanko A. D. Bilohaenko P. V. // Materials of the XX International Scientific and Practical Internet Conference "Innovations of XXI Century" Vinnitsa, 25 May, 2018. – С. 52-55.

10. Yanko A. D. Practical use cases of Bluetooth beacons on the example of IBeacon technology. / Yanko A. D. Bilohaenko P. V. // Materials of the International Scientific and Practical Internet Conference "A vector for the development of science" Vinnitsa, 20 January, 2018. – С. 5-7.

11. Yanko A. D. Building an effective deepening algorithm with tree-like information structuring for remote working and educational processes / Yanko A. D. Bilohaenko P. V. // Materials of the Abstacts of IV International Scientific and Practical Conference “Modern achievements of science and technology” Sweden, 9-10 June, 2020. – C. 51-56.