Київський національний торговельно-економічний університет
Кафедра інженерії програмного забезпечення та кібербезпеки

ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЄКТ

на тему:

«Модель інтерактивного мобільного додатку доповненої реальності інформативного спрямування»

Студента 2м курсу, 2 групи, спеціальності 121 «Інженерія програмного забезпечення» спеціалізації «Інженерія програмного забезпечення» _____ підпис студента — Галича Івана Васильовича

Науковий керівник кандидат педагогічних наук, доцент кафедри інженерії програмного забезпечення та кібербезпеки _____ підпис керівника — Жирова Тетяна Олександрівна

Гарант освітньої програми доктор економічних наук, професор кафедри інженерії програмного забезпечення та кібербезпеки _____ підпис керівника — Токар Володимир Володимирович

КИЇВ – 2021

Київський національний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення та кібербезпеки
Освітній ступінь магістр
Спеціальність 121 «Інженерія програмного забезпечення»

Затверджую
Зав. кафедри інженерії програмного забезпечення та кібербезпеки
Криворучко О. В.
«10» листопада 2020 р.

Завдання
на випускний кваліфікаційний проект студентові

Галич Іван Васильович
(прізвище, ім'я, по батькові)

1.Тема випускного кваліфікаційного проекту «Модель інтерактивного мобільного додатку доповненої реальності інформативного спрямування»

Затверджена наказом ректора від "30" листопада 2020 р. № 3224
2.Строк здачі студентом закінченого проекту 25 листопада 2021

3. Цільова установка та вихідні дані до проекту (роботи)
Мета проекту розробка системи доповненої реальності для розширення функціоналу стандартних довідкових систем
Об'єкт дослідження представлений процесом створення довідників нового покоління
Предмет дослідження - використання методу візуалізації, що базується на технологіях доповненої реальності, для створення довідників

4. Консультанти роботи із зазначенням розділів, які консультують:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв

5. Зміст випускного кваліфікаційного проекту (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. ТЕХНІЧНЕ ЗАВДАННЯ ТА МАРКЕТИНГОВЕ ДОСЛІДЖЕННЯ ДОДАТКУ

1.1. Дослідження існуючих мобільних додатків, які виконують функцію інтерактивного довідника

1.2. Загальні відомості про додаток

1.3. Мета, призначення та передумови створення інтерактивного мобільного додатку доповненої реальності

1.4. Вимоги до системи

1.5. Функціональні вимоги до додатку

Висновки до розділу 1

РОЗДІЛ 2. АНАЛІЗ ІТ ДЛЯ СТВОРЕННЯ ДОДАТКУ ДОПОВНЕННОЇ РЕАЛЬНОСТІ

2.1. Вибір рушія

2.2. Вибір мови розробки

2.3. Vuforia Augmented Reality

2.4. Microsoft Visual Studio

Висновки до розділу 2

РОЗДІЛ 3. АНАЛІЗ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ ДОДАТКУ ДОПОВНЕНОЇ РЕАЛЬНОСТІ

3.1. Аналіз варіантів використання

3.2. Анатомія та модель інтерактивного помічника

3.3. Моделювання процесів

Висновки до розділу 3

РОЗДІЛ 4. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІНТЕРАКТИВНОГО МОБІЛЬНОГО ДОДАТКУ ДОПОВНЕНОЇ РЕАЛЬНОСТІ ІНФОРМАЦІЙНОГО СПРЯМУВАННЯ

4.1. Налаштування платформи Vuforia

4.2. Початок роботи з додатком

4.3. Відстежування маркера і привітання

4.4. Робота з програмою

Висновки до розділу 4

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

6. Календарний план виконання проекту

№ пор.	Назва етапів випускного кваліфікаційного проекту	Строк виконання етапів проекту	
		за планом	фактично
1	2	3	4
1	*Вибір теми випускного кваліфікаційного проєкту*	*21.09.2020*	*21.09.2020*
2	*Розробка та затвердження завдання на проєкт магістра*	*06.11.2020*	*22.12.2020*
3	*Вступ та перелік літературних джерел*	*27.02.2021*	*27.02.2021*
4	*Вступ та перелік літературних джерел*	*20.03.2021*	*20.03.2021*
5	*Розділ 1. Технічне завдання та маркетингове дослідження додатку*	*16.04.2021*	*16.04.2021*
6	*Розділ 2. Аналіз IT для створення додатку доповненної реальності*	*24.05.2021*	*24.05.2021*
7	*Розділ 3. Аналіз інформаційних технологій для створення додатку доповненої реальності*	*21.06.2021*	*21.06.2021*
8	*Розділ 4. Реалізація та тестування інтерактивного мобільного додатку доповненої реальності інформаційного спрямування*	*20.09.2021*	*20.09.2021*
9	*Розробка програми та методики тестування*	*20.09.2021*	*20.09.2021*
10	*Написання наукової статті*	*22.05.2021*	*22.05.2021*
11	*Керівництво користувача*	*21.10.2021*	*21.10.2021*
12	*Висновки та пропозиції*	*01.11.2021*	*01.11.2021*
13	*Здача випускного кваліфікаційного проєкту на кафедру (перша перевірка)*	*03.11.2021*	*03.11.2021*
14	*Здача випускного кваліфікаційного проєкту на кафедру (перша перевірка)*	*03.11.2021*	*03.11.2021*
15	*Попередній захист випускного кваліфікаційного проєкту*	*22.11.2021 – 25.11.2021*	*22.11.2021*
16	*Здача зброшурованої випускного кваліфікаційного проєкту*	*25.11.2021*	*25.11.2021*
17	*Зовнішнє рецензування випускного кваліфікаційного проєкту*	*26.11.2021*	*26.11.2021*
18	*Підготовка до публічного захисту випускного кваліфікаційного проєкту*		

7. Дата видачі завдання «10»листопада 2020 р.

8. Науковий керівник випускного кваліфікаційного проекту Жирова Т.О.
(прізвище, ініціали, підпис)

9. Гарант освітньої програми Токар В.В.
(прізвище, ініціали, підпис)

10. Завдання прийняв до виконання студент Галич І.В.
(прізвище, ініціали, підпис)

11. Відгук керівника випускного кваліфікаційного проекту

____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____

Науковий керівник випускного кваліфікаційного проекту__________________
(підпис, дата)

Відмітка про попередній захист ____
(ПІБ, підпис, дата)

12. Висновок про випускний кваліфікаційний проект

Випускний кваліфікаційний проект студента__________Галича І.В.__________
(прізвище, ініціали)

може бути допущена до захисту екзаменаційній комісії.

Гарант освітньої програми_____________Токар В.В.________________________
(прізвище, ініціали, підпис)

Завідувач кафедри__________________Криворучко О. В.___________________
(підпис, прізвище, ініціали)

«_____»_________________20 ___ р.

АНОТАЦІЯ

Відповідно до мети дослідження робота присвячена розробці системи доповненої реальності для розширення функціоналу стандартних довідкових систем.

В результаті порівняльного аналізу аналогічних рішень було сформовано функціональні вимоги до розроблюваного додатку та описано загальну інформацію про додаток.

Розробка виконана у середовищі Unity та Visual studio 2019.

Готовий програмний комплекс Drepartment Directory було успішно протестованою.

Ключові слова: Доповнена реальность , Unity, Vuforia.

ABSTRACT

In accordance with the purpose of the study, the work is devoted to the development of an augmented reality system to expand the functionality of standard reference systems.

As a result of comparative analysis of similar solutions, the functional requirements for the developed application were formed and general information about the application was described.

Developed in Unity and Visual studio 2019

The finished software package Drepartment Directory was successfully tested.

Keywords: Augmented Reality, Unity, Vuforia

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AR - Augmented reality

VR - Virtual reality

SDK - software Development Kit

GUI - Graphical user interface

IDE - Integrated Development Environment

UML - Unified Modeling Language

ЗМІСТ

ВСТУП 5

РОЗДІЛ 1 ТЕХНІЧНЕ ЗАВДАННЯ ТА МАРКЕТИНГОВЕ ДОСЛІДЖЕННЯ ДОДАТКУ 7

1.1. Дослідження існуючих мобільних додатків, які виконують функцію інтерактивного довідника 7

1.2. Загальні відомості про додаток 13

1.3. Мета, призначення та передумови створення інтерактивного мобільного додатку доповненої реальності 13

1.4. Вимоги до системи 14

1.5. Функціональні вимоги до додатку 14

1.6 Висновки до розділу 1 15

РОЗДІЛ 2 АНАЛІЗ ІТ ДЛЯ СТВОРЕННЯ ДОДАТКУ ДОПОВНЕНОЇ РЕАЛЬНОСТІ 16

2.1. Вибір рушія 16

2.2. Вибір мови розробки 22

2.3. Vuforia Augmented Reality 24

2.4. Microsoft Visual Studio 25

2.5 Висновки до розділу 2 28

РОЗДІЛ 3 АНАЛІЗ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ ДОДАТКУ ДОПОВНЕНОЇ РЕАЛЬНОСТІ 30

3.1. Аналіз варіантів використання 30

					КНТЕУ 121– 02-03.МР		
					Модель інтерактивного мобільного додатку доповненої реальності інформативного спрямування	*Стадія*	*Аркуш*	*Аркушів*
Зм	*Аркуш*	*№ докум.*	*Підпис*	*Дата*		3	3	51
Зав. каф.		Криворучко О.В.		22.12.20				
Керівник		Жирова Т.О.		22.12.20				
Гарант		Токар В.В.		22.12.20	*Зміст*	Факультет інформаційних технологій, 2м курс, 2 група		
Розроб		Галич І.В.		22.12.20				

3.2. Анатомія та модель інтерактивного помічника 32

3.3. Моделювання процесів 37

3.4 Висновки до розділу 3 37

РОЗДІЛ 4 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІНТЕРАКТИВНОГО МОБІЛЬНОГО ДОДАТКУ ДОПОВНЕНОЇ РЕАЛЬНОСТІ ІНФОРМАЦІЙНОГО СПРЯМУВАННЯ 39

4.1. Налаштування платформи Vuforia 39

4.2. Початок роботи з додатком 46

4.3. Відстежування маркера і привітання 47

4.4. Робота з програмою 47

4.5 Висновки до розділу 4 49

ВИСНОВКИ ТА ПРОПОЗИЦІЇ 50

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 51

ДОДАТКИ 52

ВСТУП

Сучасна AR-індустрія практично повністю зайнята обслуговуванням і виробництвом контенту для мобільних пристроїв на iOS і Android. В теперішній час існує достатньо програмних засобів, призначених для вдосконалення інформаційного забезпечення даного виду у зв'язку з недосконалістю відображення різного роду довідників. 1035 додатків в App Store згадують у своєму описі, що вони використовують ARCore або ARKit для створення елементів доповненої реальності у своїй роботі. Найулюбленіша категорія в AR-розробників для iOS-пристроїв, звичайно ж, ігри (Games). Далі йде «Стиль життя» (Lifestyle) і «Продуктивність» (Productivity). «Продуктивність» – особливо популярна категорія серед користувачів iOS-пристроїв, додатки з цієї категорії допомагають людям організовувати свій робочий день, спрощують і систематизують інформацію, а також полегшують спілкування і комунікацію. Серед AR-додатків цієї категорії особливо варто відзначити: Google Translate, Mind Map AR, AR Plan 3D, AR Ruler, Torch AR, Vuforia View.

Актуальність обраної теми полягає в більш наочному способі відображення довідникової інформації. Більшість людей краще сприймає інформацію у візуальному вигляді. Однак паперовий формат все таки має деякі обмеження, які з легкістю допоможе уникнути інтерактивний довідник. За допомогою доповненої реальності, користувач, відкривши довідник, побачить не просто зображення і текст, а 3D моделі, які привертають до себе увагу і сприяють кращому

запам’ятовуванню отриманої інформації.

Мета проекту. Розробка системи доповненої реальності для розширення функціоналу стандартних довідкових систем.

З теми роботи сформовані такі *задачі:*

1. Дослідити ринок і розвиток AR в усіх сферах мобільних додатків.
2. Проаналізувати існуючі мобільні додатки подібної направленості.
3. Розглянути технологію розробки Vuforia та можливості використання.
4. Виконати проектування інтерфейсу користувача.
5. Реалізувати мобільний додаток з системою інтерактивного меню.
6. Протестувати мобільний додаток.

Об’єкт дослідження: створення довідників нового покоління.

Предмет дослідження: використання методу візуалізації, що базується на технологіях доповненої реальності для створення довідників.

Мета дослідження полягає у створенні додатку-довідника з використанням технологій доповненої реальності.

Практична цінність проекту. Отриманий результат після реалізації мобільного додатку може використовуватись в умовах кафедри як портативний інтерактивний довідник. Представлений приклад реалізації демонструє, що мобільний додаток можна використовувати в освітній сфері.

РОЗДІЛ 1
ТЕХНІЧНЕ ЗАВДАННЯ ТА МАРКЕТИНГОВЕ ДОСЛІДЖЕННЯ ДОДАТКУ

1.1. Дослідження існуючих мобільних додатків, які виконують функцію інтерактивного довідника

В сучасному світі технології доповненої реальності є надзвичайно популярними, через що кількість продуктів, які можна вважати інтерактивними довідниками, надзвичайно велика, через це слід розглянути приклади із різних сфер.

Найпопулярніший додаток-довідник в освітній сфері: Narrator AR.

Narrator AR – це додаток доповненої реальності, який використовується для навчання дітей письму.

Додаток поставляється в комплекті з спеціальним зошитом-прописами, на сторінках якого розміщені маркери, які призводять до активації алгоритмів доповненої реальності додатку.

Приклади інтерфейсу зображено на рисунках 1.1 – 1.3.

На рисунку 1.1 зображено приклад інтерфейсу додатку з розпізнаванням рамки завдання.

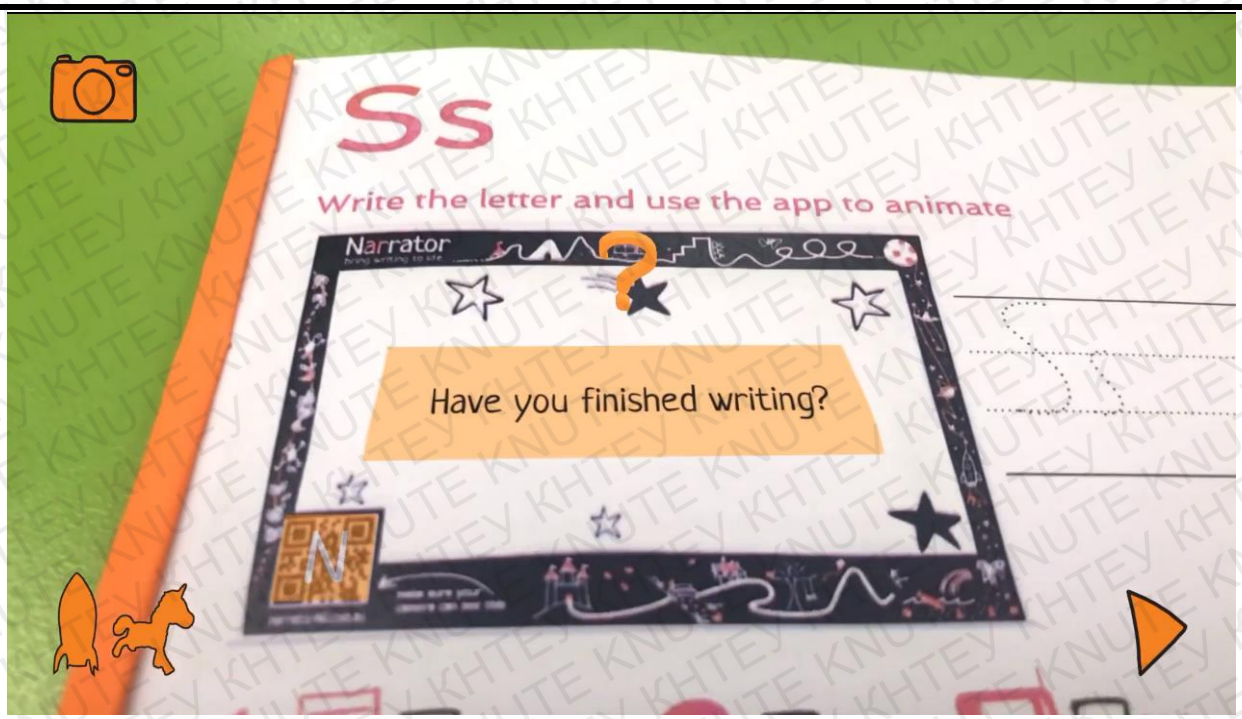


Рис. 0.1 Приклад інтерфейсу Narrator AR (1)

На рисунку 1.2 зображено приклад розпізнавання QR-коду на листі завдання в додатку Narrator AR.



Рис. 0.2 Приклад інтерфейсу Narrator AR (2)

На рисунку 1.3 представлено розпізнавання QR-коду на листі завдання в додатку Narrator AR.

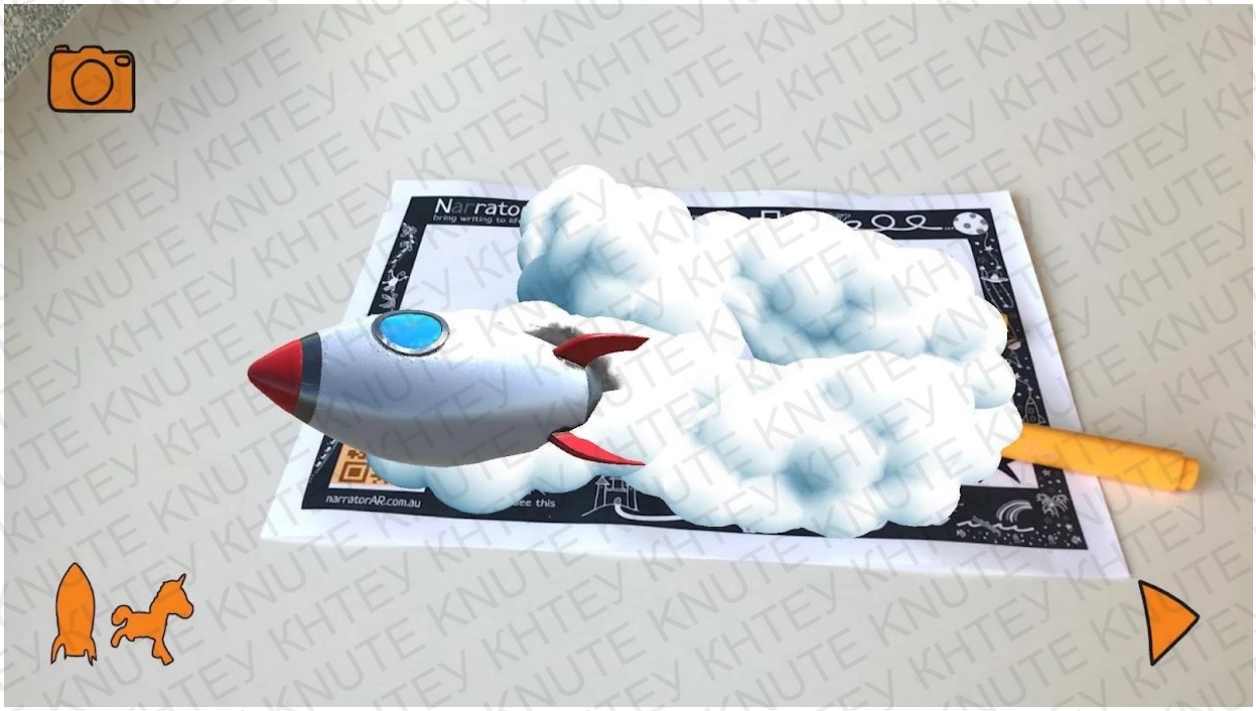


Рис. 0.3 Приклад інтерфейсу Narrator AR (3)

Даний додаток представляє багатий функціонал і велику кількість різних 3D моделей, виконаних в стилі мультфільмів, що сприяє підвищенню сприйняття інформації дітьми, на яких розраховано додаток.

Окрім освітньої сфери довідники з використанням доповненої реальності не обійшли стороною і сферу ресторанного бізнесу.

На сьогоднішній день багато ресторанів і кафе впроваджують меню в доповненій реальності, що дозволяю по-новому поглянути на звичайний перелік страв.

Одним з найяскравіших прикладів можна вважати додаток Menu AR (Посилання на додаток – [1]). Даний додаток дозволяє переглянути існуючі в меню страви у вигляді 3D моделей, виконаних в реальних розмірах.

Приклади інтерфейсу зображені на рисунках 1.4 – 1.7.

На рисунку 1.4 представлено скріншот із меню з використанням доповненої реальності в портретному форматі.



Рис. 0.4 Приклад інтерфейсу Menu AR (1)

На рисунку 1.5 зображено приклад інтерфейсу без додаткових панелей опису і функціональних кнопок.



Рис. 0.5 Приклад інтерфейсу Menu AR (2)

На рисунку 1.6 представлений приклад інтерфейсу з відображенням всіх функціональних панелей додатку.



Рис. 0.6 Приклад інтерфейсу Menu AR (3)

На рисунку 1.7 зображено інтерфейс додатку в пейзажному режимі.

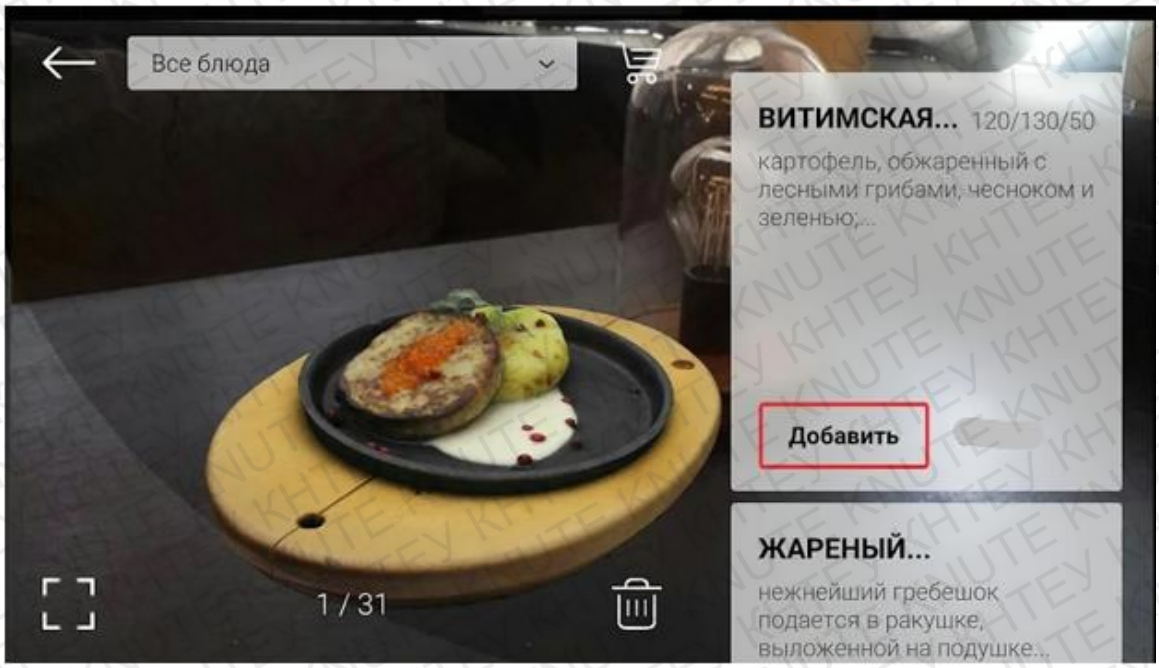


Рис. 0.7 Приклад інтерфейсу Menu AR (4)

В загальному сенсі суть додатків є майже однаковою – представлення інформації, яка зазвичай подається у друкованому вигляді, у більш зрозумілому і сприйнятому для розуміння форматі 3D візуалізацій з використанням технологій доповненої реальності.

1.2. Загальні відомості про додаток

Додаток має представляти собою інтерактивний довідник, який покликаний дати загальний огляд освітніх програм кафедри «Інженерія програмного забезпечення та кібербезпеки» та може використовуватися у профорієнтаційних цілях.

Основний функціонал додатку буде полягати в використанні технологій доповненої реальності і 3D візуалізацій з метою покращення якості сприймання інформації.

Додаток представляє собою систему-довідник, в якій форма подачі заздалегідь збереженої інформації полягає в «розповіді» від 3D моделі, яку можна побачити виключно через камеру смартфона, інтегрованою у реальний світ.

1.3. Мета, призначення та передумови створення інтерактивного мобільного додатку доповненої реальності

Основна мета створення інтерактивного мобільного додатку доповненої реальності кафедри «Інженерія програмного забезпечення та кібербезпеки» полягає в поліпшенні способу передачі інформації стосовно освітніх програм кафедри.

Передумовами можна вважати власне розуміння складності пошуку і сприйняття необхідної інформації стосовно освітніх програм кафедри і загального

розуміння її призначення, засноване на власному досвіді, отриманому при вступі до вузу.

Призначення додатку полягає в актуалізації і діджиталізації довідникової інформації, шо призведе до:

1. Поліпшення сприйняття інформації користувачами довідника
2. Підвищення зацікавленості кафедрою за рахунок нестандартного підходу до побудови довідника

1.4. Вимоги до системи

Довідник розробляється для кінцевих користувачів, які використовують операційну систему Android OS, тож основні вимогу до системи такі:

- операційна система Android 4.4.2 (KitKat) і старше;
- 200 мегабайт вільного місця на пристрої;
- функціонуюча камера, не менше 4 мегапілселів;
- наявність ліхтарика камери для використання в темний час доби;
- увімкнена функція «Встановлення з невідомих джерел».

1.5. Функціональні вимоги до додатку

Основний функціонал, який має виконувати додаток:

- привітання користувача;
- функція вибору потрібної освітньої програми;
- передача повної інформації про освітні програми в доступному форматі;
- можливість повернутися до вибору освітньої програми;
- для виборів використовуються віртуальні кнопки;
- загальний розмір файлу-інсталятора – не більший за 200 мегабайт;

- повноцінна взаємодія з додатком через камеру;
- час на відстеження мітки – не більше 5-6 секунд;
- затримки у відгуку на натискання віртуальної кнопки – недопустимі.

1.6 Висновки до розділу 1

В даному розділі проведено огляд основних аналогів і рішень в обраній сфері, розглянуто їх структуру, інтерфейс і призначення.

На базі інформації, отриманої в результаті аналізу аналогів, було сформовано функціональні вимоги до розроблюваного додатку та описано загальну інформацію про додаток.

Окрім цього було визначено мету, передумови та призначення розробки, описано вимоги до операційної системи кінцевого користувача.

Вся отримана інформація є основою для подальшої розробки інтерактивного мобільного додатку з використанням технологій доповненої реальності.

На основі першого розділу можна зробити висновок про абсолютну необхідність створення подібного роду додатку, адже це підвищить якість сприйняття інформації стосовно освітніх програм кафедри, що в свою чергу посприяє:

- підвищенню інтересу до кафедри в цілому;
- більшій усвідомленості абітурієнтів і студентів стосовно освітніх програм.

РОЗДІЛ 2
АНАЛІЗ ІТ ДЛЯ СТВОРЕННЯ ДОДАТКУ ДОПОВНЕНОЇ РЕАЛЬНОСТІ

2.1. Вибір рушія

Unity – це міжплатформенний ігровий рушій, розроблений Unity Technologies, вперше оголошений та випущений у червні 2005 року на Всесвітній конференції розробників Apple Inc. як ігровий рушій Mac OS X. Станом на 2018 рік, рушій був розширений для підтримки понад двадцять п’ять платформ. Рушій може бути використаний для створення тривимірних, двовимірних, віртуальної реальності та ігор з доповненою реальністю, а також моделювання та іншого. Рушій був прийнятий у галузях, що не входять до відеоігор, таких як кіно, автомобілебудування, архітектура, інженерія та будівництво.

З моменту запуску було випущено кілька основних версій Unity. Остання стабільна версія – 2019.3.6, була випущена в березні 2020 року.

Ігровий рушій Unity, запущений у 2005 році, спрямований на «демократизацію» розвитку ігор, зробивши її доступною для більшої кількості розробників [4,5]. Наступного року Unity був названий переможцем у категорії «Найкраще використання графіки Mac OS X» в Apple Design Awards Apple Inc [6]. Спочатку Unity був випущений для Mac OS X, пізніше додав підтримку для Microsoft Windows та веб-браузерів [7].

Реліз Unity 2.0 пройшов 2007 році та додав до існуючого функціоналу приблизно 50 нових можливостей [8]. Випуск включав оптимізований рушій місцевості для детальних 3D-середовищ, динамічних тіней у режимі реального часу, спрямованих вогнів та прожекторів, відтворення відео та інших функцій [8]. Випуск також додав функції, завдяки яким розробники могли співпрацювати легше. Він включав мережевий рівень для розробників для створення багатокористувацьких ігор на основі протоколу User Datagram, що пропонує переклад мережевих адрес, синхронізацію стану та виклики віддалених процедур [8].

Коли Apple запустила свій App Store у 2008 році, Unity швидко додав підтримку iPhone. Кілька років цей рушій був беззаперечним на iPhone, і він став добре відомий розробникам ігор для iOS [7].

Unity 3.0 запущений у вересні 2010 року з функціями, що розширюють графічні можливості рушія для настільних комп'ютерів та консолей відеоігор [10]. Окрім підтримки Android, Unity 3 передбачає інтеграцію інструменту Beast Lightmap Illuminate Labs 'Beast Lightmap, відкладену візуалізацію, вбудований редактор дерев, власне відображення шрифтів, автоматичне ультрафіолетове відображення та аудіофільтри.

Опитування журналу Game Developer у травні 2012 року вказало Unity як головний ігровий движок для мобільних платформ . У листопаді 2012 року Unity Technologies поставила Unity 4.0. Ця версія додала підтримку DirectX 11 та Adobe Flash, нові засоби анімації під назвою Mecanim та доступ до попереднього перегляду Linux.

Facebook інтегрував комплект розробки програмного забезпечення для ігор за допомогою ігрового рушійа Unity у 2013 році. Тут представлені інструменти, що дозволяють відстежувати рекламні кампанії та здійснювати глибоке посилання, де користувачі безпосередньо пов’язувались із публікацій у соціальних мережах до

певних частин в іграх, а також легко ділитися зображеннями в грі. У 2016 році Facebook розробила нову ігрову платформу для ПК з Unity. Unity надав підтримку ігровим платформам Facebook, а розробники Unity могли швидше експортувати та публікувати ігри у Facebook.

У Verge сказано про випуск Unity 5 2015 року: «Unity почався з того, щоб зробити розробку гри загальнодоступною. Unity 5 – довгоочікуваний крок до цього майбутнього». З Unity 5 рушій удосконалився, його освітлення та аудіо [17]. Через WebGL розробники Unity могли додавати свої ігри до сумісних веб-браузерів, не потребуючи плагінів для гравців. Unity 5.0 пропонував глобальне освітлення в режимі реального часу, попередній перегляд світлого відображення, Cloud Cloud, нову аудіосистему та фізичний рушій Nvidia PhysX 3.3. П’яте покоління рушія Unity також представило ефекти Cinematic Image Effects, щоб зробити ігри Unity менш загальними. Unity 5.6 додав нові ефекти освітлення та частинок, оновив загальну продуктивність рушія та додав підтримку Nintendo Switch, Facebook Gameroom, VR Daydream VR та графічного API Vulkan [19]. Він представив 4K відеоплеєр, здатний запускати 360-градусні відеоролики для віртуальної реальності. Однак деякі геймери критикували доступність Unity через великий обсяг швидко виготовлених ігор, опублікованих на платформі дистрибуції Steam недосвідченими розробниками.

У грудні 2016 року Unity Technologies оголосили, що змінять систему нумерації версій для Unity з ідентифікаторів, заснованих на послідовності, до року випуску, щоб вирівняти версію з їх більш частою каденцією випуску; Отже, за Unity 5.6 пішов Unity 2017. Інструменти Unity 2017 демонстрували рушій візуалізації в режимі реального часу, класифікацію кольорів та світобудування, аналітику операцій та звітність про ефективність. Unity 2017.2 підкреслив плани Unity Technologies за межами відеоігор. Сюди входили нові інструменти, такі як Timeline, що дозволяло розробникам перетягувати анімацію в ігри, і Cinemachine, розумна система камер в іграх. Unity 2017.2 також інтегрував інструменти 3DS Max і Maya

Autodesk в рушій Unity для спрощеного процесу обміну активами під час ітерації ігор.

Unity 2018 показав розробникам новий шаблон Scriptable Rende Pipeline, призначений для створення графіки високого класу. Сюди входили конвеєр рендерингу високої чіткості для консольних та ПК, а також легкий конвеєр рендерингу для мобільних, віртуальної реальності, доповненої реальності та змішаної реальності. Unity 2018 також включав інструменти машинного навчання, такі як Imitation Learning, за допомогою яких ігри вчаться з реальних звичок гравця, підтримка Magic Leap та шаблони для нових розробників.

У 2019 році було додано нове посилання на мову Wolfram, що дозволяє отримати доступ до функцій високого рівня мови Wolfram від Unity. Виклик об'єктів Unity з мови Wolfam також став можливим через бібліотеку UnityLink.

Unity — це кросплатформний рушій. Редактор Unity підтримується на платформі Windows, macOS та Linux, тоді як сам рушій наразі підтримує створення ігор для більш ніж 19 різних платформ, включаючи мобільні, настільні, консолі та віртуальну реальність. Офіційно підтримувані платформи з Unity 2020 LTS:

- Мобільні платформи iOS, Android (Android TV), tvOS;
- Настільні платформи Windows (Універсальна платформа Windows), Mac, Linux;
- Веб-платформа WebGL;
- Консольні платформи PlayStation (PS4, PS5), Xbox (Xbox One, Xbox Series X/S), Nintendo Switch, Stadia;
- Платформи віртуальної/розширеної реальності Oculus, PlayStation VR, Google ARCore, Apple ARKit, Windows Mixed Reality (HoloLens), Magic Leap та через Unity XR SDK Steam VR, Google Cardboard.

Раніше підтримуваними платформами були Wii, Wii U, PlayStation 3, Xbox 360, Tizen, PlayStation Vita, 3DS, BlackBerry 10, Windows Телефон 8, Samsung Smart TV, Gear VR, Daydream, Vuforia і Facebook Gameroom.

Модель ліцензування

Протягом перших десяти років як продукту, платні версії Unity були продані повністю; у 2016 році корпорація перейшла на модель передплати. Unity має безкоштовні та платні варіанти ліцензування. Безкоштовна ліцензія призначена для особистого користування або для невеликих компаній, які заробляють менше 100 000 доларів США на рік, пізніше збільшено до 200 000 доларів США, а підписки засновані на доходах, отриманих від ігор за допомогою Unity. Платний варіант, Unity Pro, був необхідний для розробників, які мали понад 200 000 доларів США річного доходу, але його також можна було надати розробникам консолей через ліцензію на перевагу платформи від виробника консолі. Ключі Unity Pro були частиною іншого SDK від виробника консолі, за який розробник заплатив. У червні 2021 року Unity дещо змінила цей план, щоб будь-який розробник, який створює ігри на закритих консольних системах (PlayStation, Nintendo Switch і Xbox), незалежно від доходу, мав ліцензію Unity Pro або ліцензійний ключ пріоритетної платформи від виробників. Sony і Nintendo надають це як частину SDK, але Microsoft ще не впровадила цю функцію для свого SDK.

Вихідний код двигуна ліцензується на основі спеціального договору.

Магазин активів Unity

Творці можуть розробляти та продавати створені користувачами активи іншим виробникам ігор через Unity Asset Store. Сюди входять 3D- та 2D-активи та середовища, які розробники можуть купувати та продавати. Unity Asset Store був запущений у 2010 році. До 2018 року через цифровий магазин було завантажено приблизно 40 мільйонів.

Станом на 2018 рік Unity використовувався для створення приблизно половини мобільних ігор на ринку та 60 відсотків вмісту доповненої реальності та

віртуальної реальності, у тому числі приблизно 90 відсотків на нових платформах доповненої реальності, таких як Microsoft HoloLens, і 90 відсоток вмісту Samsung Gear VR. Технологія Unity є основою для більшості випадків віртуальної реальності та доповненої реальності, і Fortune сказав, що Unity «домінує у бізнесі віртуальної реальності». Unity Machine Learning Agents — це програмне забезпечення з відкритим кодом, завдяки якому платформа Unity підключається до програм машинного навчання, включаючи TensorFlow від Google. Використовуючи метод проб і помилок в Unity Machine Learning Agents, віртуальні персонажі використовують навчання з підкріпленням для створення творчих стратегій у реальних віртуальних ландшафтах. Програмне забезпечення використовується для розробки роботів і самокерованих автомобілів.

Раніше Unity підтримував інші платформи, включаючи власний веб-програвач Unity, плагін для веб-браузера. Однак він був застарілий на користь WebGL. Починаючи з версії 5, Unity пропонує свій пакет WebGL, скомпільований на JavaScript, використовуючи 2-етапний мовний перекладач (C# на C++ і, нарешті, на JavaScript).

Unity був стандартним комплектом для розробки програмного забезпечення (SDK), який використовувався для ігрової консолі Wii U від Nintendo, з безкоштовною копією, яка включена Nintendo до кожної ліцензії розробника Wii U. Unity Technologies назвала це комплектування стороннього SDK «першою галуззю».

Unity надає користувачам можливість створювати ігри та переживання як у 2D, так і в 3D, а рушій пропонує первинний API сценарію в C#, як для редактора Unity у вигляді плагінів, так і для самих ігор, а також функцій перетягування. До того, як C# була основною мовою програмування, що використовується для рушія, вона раніше підтримувала Boo, який було видалено з випуском Unity 5, та версією JavaScript під назвою UnityScript, яка була застаріла в серпні 2017 року після виходу Unity 2017.1, на користь C#.

У межах 2D ігор Unity дозволяє імпортувати спрайтів та просунутий 2D світовий рендер. Для 3D-ігор Unity дозволяє задавати стиснення текстури, налаштування роздільної здатності для кожної платформи, яку підтримує ігровий рушій, і забезпечує підтримку збільшеного картографування, відображення відображень, картографування паралакса, оклюзії навколишнього простору екрану (SSAO), динамічного тіні, використовуючи тіньові карти, зображення текстури та ефекти після повної обробки на екрані.

Станом на 2018 рік Unity використовувався для створення приблизно половини нових мобільних ігор на ринку та 60 відсотків змісту доповненої реальності та віртуальної реальності.

2.2. Вибір мови розробки

C# – це універсальна багатопарадігмальна мова програмування, що включає строгу типізацію, лексичну область видимості, імператив, декларативну, функціональну, універсальну об'єкто-орієнтовану (на основі класів) та компонентно-орієнтовані дисципліни програмування [1]. C# була розроблена Microsoft приблизно в 2000 році в рамках ініціативи .NET, а потім затверджена в якості міжнародного стандарту Ecma (ECMA-334) і ISO (ISO / IEC 23270: 2018). Mono – це назва безкоштовного проекту з відкритим вихідним кодом для розробки компілятора і середовища виконання для мови. C# є однією з мов програмування, розроблених для Common Language Infrastructure (CLI).

C# був розроблений Андерсом Хейлсбергом, а його командою розробників в даний час керує Мадс Торгерсен. Остання версія – 9.0, випущена в 2021 році [2, 3].

Під час розробки .NET Framework бібліотеки класів спочатку були написані з використанням компілятора системи керованого коду під назвою «Simple Managed C» (SMC). У січні 1999 року Андерс Хейлсберг сформував команду для створення нової мови під назвою Cool, який позначав «C-подібна об'єктно-

орієнтована мова». Microsoft розглядала збереження назви «Cool» в якості остаточної назви мови, але вирішила не робити цього з причин, пов'язаних з товарними знаками. До того часу, коли проект .NET був публічно оголошено на конференції професійних розробників в липні 2000 року, мову був перейменований в C#, а бібліотеки класів і середовище виконання ASP.NET були портовані на C#.

Хейлсберг – головний дизайнер і головний архітектор C# в Microsoft, раніше займався проектуванням Turbo Pascal, Embarcadero Delphi (раніше CodeGear Delphi, Inprise Delphi і Borland Delphi) і Visual J ++. У своїх інтерв'ю і технічних статтях він заявив, що недоліки в більшості основних мов програмування (наприклад, C ++, Java, Delphi і Smalltalk) призвели до основ Common Language Runtime (CLR), що, в свою чергу, призвело до розробки саме мови C#.

Джеймс Гослінг, який створив мову програмування Java в 1994 році, і Білл Джой, співзасновник Sun Microsystems, творця Java, назвали C# «імітацією» Java; Далі Гослінг сказав, що «C# - це свого роду Java з надійністю, продуктивністю і безпекою видалений». Клаус Крефт і Анджеліка Лангер (автори книги про потоках C ++) заявили у своєму блозі, що «Java і C#» це майже ідентичні мови програмування. Нудне повторення без інновацій», «Навряд чи хто-небудь буде стверджувати, що Java або C# є революційними мовами програмування, які змінили спосіб написання програм», а «C# багато запозичила у Java - і навпаки.

По своєму дизайну C# є мовою програмування, який безпосередньо відображає основну інфраструктуру спільної мови (CLI). Більшість його внутрішніх типів відповідають типам значень, реалізованих середовищем CLI. Однак специфікація мови не містить вимог до генерації коду компілятора, тобто не вказує, що компілятор C# має орієнтуватися на середовище виконання спільної мови, або генерувати загальний проміжний мова (CIL), або генерувати будь-який інший конкретний формат. Теоретично, компілятор C# може генерувати машинний код, як традиційні компілятори C ++ та Fortran.

C# підтримує строго типізовані оголошення неявних змінних з ключовим словом var, а також неявно типізовані масиви з ключовим словом new [], за яким слідує ініціалізатор колекції.

C# більш безпечна для типів, ніж C++. Єдиними неявними перетвореннями за замовчуванням є ті, які вважаються безпечними, наприклад, розширення цілих чисел. Це застосовується під час компіляції, під час JIT і, в деяких випадках, під час виконання. Не відбувається неявного перетворення між логічними значеннями і цілими числами, а також між членами перерахування і цілими числами (за винятком літерала 0, який може бути неявно перетворений в будь-який тип, що перераховується). Будь-яке кероване перетворення повинне бути чітко позначено як явне або неявне, на відміну від конструкторів копіювання C++ та операторів перетворення, які за замовчуванням неявні.

2.3. Vuforia Augmented Reality

Vuforia – це набір для розробки програмного забезпечення розширеної реальності (SDK) для мобільних пристроїв, що дозволяє створювати додатки розширеної реальності. Він використовує технологію комп'ютерного зору для розпізнавання та відстеження планарних зображень та 3D-об'єктів у режимі реального часу. Ця можливість реєстрації зображень дозволяє розробникам розміщувати та орієнтувати віртуальні об'єкти, такі як 3D-моделі та інші медіа, стосовно об'єктів реального світу, коли вони переглядаються через камеру мобільного пристрою. Потім віртуальний об'єкт відстежує положення та орієнтацію зображення в режимі реального часу, щоб погляд глядача на об'єкт відповідав перспективі на ціль. Таким чином, виявляється, що віртуальний об’єкт є частиною сцени реального світу.

Пакет SDK Vuforia підтримує різноманітні 2D та 3D-цілі, включаючи цілі зображення без маркерів, цільову модель 3D та форму адресованого Fiducial

Marker, відомого як VuMark. Додаткові функції SDK включають 6 ступенів локалізації пристрою свободи в просторі, локалізоване виявлення оклюзії за допомогою «Віртуальних кнопок», вибір цілі зображення під час виконання та можливість створення та переналаштування наборів цілей програмно під час виконання програми.

Vuforia надає інтерфейси прикладного програмування (API) на мовах C ++, Java, Objective-C ++ та .NET через розширення до ігрового механізму Unity. Таким чином, SDK підтримує як вбудовану розробку для iOS, Android, так і UWP, в той час як він також дозволяє розробляти AR-програми в Unity, які легко переносяться на обох платформах.

2.4. Microsoft Visual Studio

Microsoft Visual Studio – це інтегроване середовище розробки (IDE) від Microsoft. Він використовується для розробки комп'ютерних програм, а також веб-сайтів, веб-додатків, веб-служб та мобільних додатків. Visual Studio використовує платформи Microsoft для розробки програмного забезпечення, такі як API Windows, Windows Forms, Foundation Presentation Foundation, Windows Store та Microsoft Silverlight. Він може створювати як власний код, так і керований код.

Visual Studio включає редактор коду, що підтримує IntelliSense (компонент доповнення коду), а також рефакторинг коду. Інтегрований налагоджувач працює як налагоджувач на рівні джерела, так і на рівні машини. Інші вбудовані інструменти включають, конструктор для побудови програм GUI, веб-дизайнер, дизайнер класів та дизайнер схем бази даних. Він приймає плагіни, які покращують функціональність майже на кожному рівні - включаючи додавання підтримки для систем керування джерелами (наприклад, Subversion і Git) та додавання нових наборів інструментів, таких як редактори та візуальні дизайнери для мов, що

задаються доменом або набори інструментів для інших аспектів розробки програмного забезпечення життєвий цикл (як клієнт Azure DevOps: Team Explorer).

Visual Studio підтримує 36 різних мов програмування та дозволяє редактору коду та налагоджувачу підтримувати (в різній мірі) майже будь-яку мову програмування за умови існування послуги, що залежить від мови. Вбудовані мови включають C, C ++, C ++ / CLI, Visual Basic .NET, C#, F #, JavaScript, TypeScript, XML, XSLT, HTML та CSS. Підтримка інших мов, таких як Python, Ruby, Node.js та M серед інших, доступна через плагіни. Java (і J #) підтримувалися в минулому.

Найбільш основне видання Visual Studio, спільноти, доступне безкоштовно. Гасло для видання Visual Studio Community - "Безкоштовне повнофункціональне IDE для студентів, відкритих джерел та індивідуальних розробників".

На даний момент підтримується версія Visual Studio - 2019 рік.

Visual Studio не підтримує жодної мови програмування, рішення чи інструменту внутрішньо; натомість це дозволяє підключати функціональність, кодовану як VSPackage. Після встановлення функціональність доступна як Сервіс. IDE надає три послуги: SVsSolution, яка надає можливість перераховувати проекти та рішення; SVsUIShell, який забезпечує функцію вікон та інтерфейсу користувача (включаючи вкладки, панелі інструментів та вікна інструментів); та SVsShell, яка займається реєстрацією VSPackages. Крім того, IDE також відповідає за координацію та забезпечення зв'язку між службами. Всі редактори, дизайнери, типи проектів та інші інструменти реалізовані як VSPackages. Visual Studio використовує COM для доступу до VSPackages. SDK Visual Studio також включає в себе керовану рамку пакетів (MPF), яка є набором керованих обгорток навколо COM-інтерфейсів, які дозволяють писати пакети будь-якою мовою, сумісною з CLI. Однак MPF не забезпечує всю функціональність, що піддається інтерфейсам Visual Studio COM. Потім послуги можуть бути використані для створення інших пакетів, які додають функціональність Visual Studio IDE.

Підтримка мов програмування додається за допомогою спеціального VSPackage, який називається Language Service. Мовна служба визначає різні інтерфейси, які може реалізувати реалізація VSPackage, щоб додати підтримку різних функціональних можливостей. Функції, які можна додати таким чином, включають забарвлення синтаксису, завершення оператора, відповідність дужок, підказки інформації про параметри, списки членів та маркери помилок для складання фону. Якщо інтерфейс буде реалізований, функціонал буде доступний для мови. Мовні послуги реалізуються на мовній основі. Реалізації можуть повторно використовувати код з аналізатора або компілятора для мови. Мовні сервіси можуть бути реалізовані як у рідному коді, так і керованому коді. Для нативного коду можуть бути використані або рідні інтерфейси COM, або Babel Framework (частина Visual Studio SDK). Для керованого коду MPF включає обгортки для написання сервісів керованої мови.

Visual Studio не включає вбудовану підтримку управління джерелами, але вона визначає два альтернативних способи інтеграції систем управління джерелами з IDE. VSPackage управління джерелом може забезпечити власний індивідуальний інтерфейс користувача. Навпаки, плагін управління джерелом за допомогою MSSCCI (Microsoft Source Code Control Interface) надає набір функцій, які використовуються для реалізації різних функцій управління джерелом, зі стандартним інтерфейсом користувача Visual Studio. MSSCCI вперше використовувався для інтеграції Visual SourceSafe з Visual Studio 6.0, але згодом був відкритий через SDK Visual Studio. Visual Studio .NET 2002 використовував MSSCCI 1.1, а Visual Studio .NET 2003 використовували MSSCCI 1.2. Visual Studio 2005, 2008 та 2010 використовує MSSCCI версії 1.3, яка додає підтримку перейменування та видалення розповсюдження, а також асинхронного відкриття.

Visual Studio підтримує запуск декількох екземплярів середовища (кожен зі своїм набором VSPackages). Екземпляри використовують різні вулики реєстру (див. Визначення MSDN терміна "вулик реєстру" у сенсі, що використовується тут)

для зберігання конфігураційного стану та диференціюються їх AppId (ідентифікатор програми). Екземпляри запускаються специфічним для AppId .exe, який вибирає AppId, встановлює кореневий вулик та запускає IDE. VSP-пакети, зареєстровані для одного AppId, інтегруються з іншими VSP-пакетами для цього AppId. Різні випуски продуктів Visual Studio створені за допомогою різних додатків. Продукти випуску Visual Studio Express встановлюються разом із власними AppIds, але продукти Standard, Professional та Team Suite мають однаковий AppId. Отже, видання Express можна встановити поряд з іншими виданнями, на відміну від інших видань, які оновлюють ту саму установку.

Професійне видання включає в себе набір VSPackages у стандартному виданні, а командний набір включає суперсет VSPackages в обох інших виданнях. Система AppId використовується Visual Studio Shell в Visual Studio 2008.

Дане середовище розробки було обрано по декількох критеріях:

- Підтримка мови C#.
- Інтуїтивно зрозумілий інтерфейс.
- Можливості графічного редактору, підходящі до задач.

Так як усі критерії задовільнено – середовищем розробки було обрано саме Visual Studio 2019.

2.5 Висновки до розділу 2

В даному розділі проведено огляд основних інструментів для створення додатку доповненої реальності.

На базі інформації, отриманої в результаті аналізу інструментів, було вибрано інструменти для розробки додатку.

Unity пропонує потужні інструменти для розробки насичених та захоплюючих додатків у доповненій реальності з інтелектуальною взаємодією з реальним світом.

Вся отримана інформація є основою для подальшої розробки інтерактивного мобільного додатку з використанням технологій доповненої реальності.

Отже, Unity пропонує потужні інструменти для розробки насичених та захоплюючих додатків у доповненій реальності з інтелектуальною взаємодією з реальним світом, Vuforia – це набір для розробки програмного забезпечення розширеної реальності (SDK) для мобільних пристроїв, що дозволяє створювати додатки розширеної реальності.

РОЗДІЛ 3
АНАЛІЗ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ ДОДАТКУ ДОПОВНЕНОЇ РЕАЛЬНОСТІ

3.1. Аналіз варіантів використання

Діаграма використання найпростіша – це представлення взаємодії користувача із системою, яка показує взаємозв'язок між користувачем та різними випадками використання, в яких користувач бере участь. Діаграма випадків використання може ідентифікувати різні типи користувачів системи та різні випадки використання, і часто вона супроводжується також іншими типами діаграм. Варіанти використання представлені кругами або еліпсами.

Незважаючи на те, що сам випадок використання може детально вивчити кожну можливість, діаграма прикладів використання може допомогти забезпечити огляд системи на більш високому рівні. Раніше вже було сказано, що "схеми використання - це принципи вашої системи".

Через їх спрощений характер, схеми використання можуть бути хорошим інструментом комунікації для зацікавлених сторін. Креслення намагаються імітувати реальний світ і дають зацікавленій стороні уявлення про те, як буде розроблена система. Сіау та Лі провели дослідження, щоб визначити, чи взагалі існувала дійсна ситуація для схем використання або вони були непотрібними. Було виявлено, що діаграми випадків використання передають намір системи більш спрощеним чином зацікавленим сторонам і що вони "інтерпретуються більш повно,

ніж діаграми класів".

Метою діаграми використання є відображення динамічного аспекту системи. Додаткові схеми та документація можуть бути використані для забезпечення повного функціонального та технічного уявлення про систему. Вони забезпечують спрощене та графічне представлення того, що система насправді повинна робити.

Елементи:

- рамки системи (англ. system border) - прямокутник із назвою у верхніх частинах та еліпсами (прецедентами) всередині. Часто може бути опущено без корисної інформації про корисну інформацію,
- актор (англ. актор) - стилізований людський персонаж, позначаючи набір ролей користувача (розуміється в широкому змісті: людина, зовнішня сутність, клас, інша система), взаємодіючого з деякою сутністю (системною, підсистемою, класом). Актори не можуть бути пов'язані між собою з іншим (за вимкнення відносин щодо обробки / дослідження),
- прецедент - еліпс із надписом, що позначає виконувану систематичну дію (може включати можливі варіанти), що призводить до спостерігаємих акторами результатів. Надпис може бути ім'ям або описом (з точки зору актора) того, "що" робить система (а не "як"). Ім'я прецедента зв’язано з неперервним (атомарним) сценарієм - конкретною послідовністю дій, ілюструючи подію. Під час сценарію актори обмінюються із систематичними повідомленнями. Сценарій може бути приведений на діаграмі прецедентів у відео UML-коментарі. З одним прецедентом може бути пов'язано кілька різних сценаріїв

На рисунку 3.1 зображено діаграму варіантів використання, яка описує можливі дії користувача в системі.

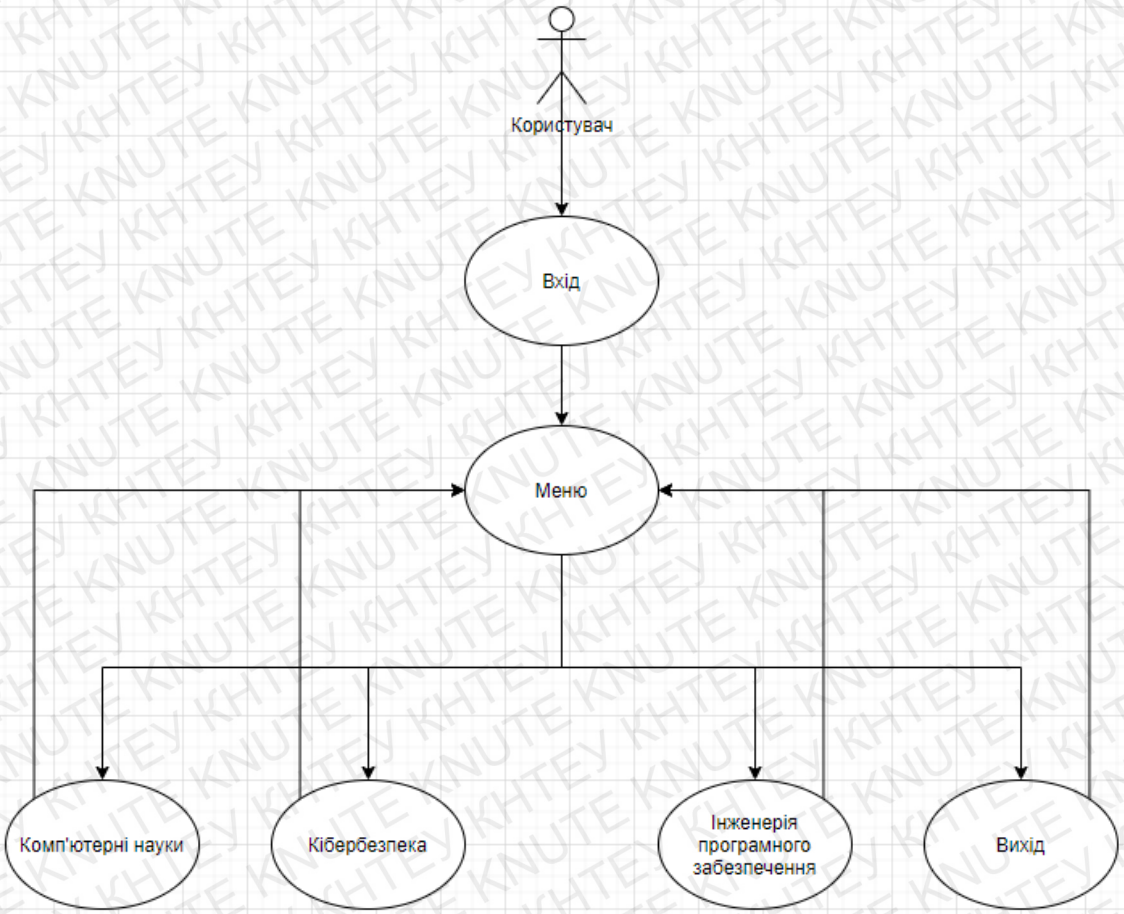


Рис. 0.1 Діаграма варіантів використання

3.2. Анатомія та модель інтерактивного помічника

У програмній інженерії діаграма класів в Уніфікованій мові моделювання (UML) - це тип статичної структурної діаграми, що описує структуру системи, показуючи класи системи, їх атрибути, операції (або методи) та взаємозв'язки між об'єктами.

Діаграма класів є основним будівельним елементом об’єктно-орієнтованого моделювання. Він використовується для загального концептуального моделювання

структури програми та для детального моделювання переведення моделей у програмовий код. Діаграми класів також можуть бути використані для моделювання даних. Класи на діаграмі класів представляють як основні елементи, взаємодії в програмі, так і класи, що програмуються.

На схемі класи представлені вікнами, які містять три відділення:

- У верхньому відділенні міститься назва класу. Надруковано жирним шрифтом і відцентровано, а перша літера написана великими літерами.
- Середній відсік містить атрибути класу. Вони вирівняні за лівим краєм, а перша буква мала.
- У нижньому відділенні містяться операції, які може виконувати клас. Вони також вирівняні за лівим краєм, а перша буква - мала.

При проектуванні системи ряд класів ідентифікується та згруповується у схему класів, яка допомагає визначити статичні відносини між ними. При детальному моделюванні класи концептуального проекту часто поділяються на ряд підкласів.

Залежність - це семантичний зв'язок між залежними та незалежними елементами моделі. Він існує між двома елементами, якщо зміни у визначенні одного елемента (сервера або цілі) можуть спричинити зміни для іншого (клієнта або джерела). Ця асоціація є односпрямованою. Залежність відображається у вигляді штрихової лінії з відкритою стрілкою, яка вказує від клієнта до постачальника.

Для подальшого опису поведінки систем ці діаграми класів можуть бути доповнені діаграмою стану або машиною стану UML.

Асоціація представляє родину посилань. Двійкова асоціація (з двома кінцями) зазвичай представляється у вигляді рядка. Асоціація може пов'язувати будь-яку кількість класів. Асоціація з трьома ланками називається потрійною асоціацією. Асоціацію можна назвати, а кінці асоціації можна прикрасити іменами ролей, показниками власності, кратністю, видимістю та іншими властивостями.

Існує чотири різні типи асоціацій: двонаправлена, односпрямована, агрегаційна (включає агрегацію композиції) та рефлексивна. Двонаправлені та односпрямовані асоціації є найбільш поширеними.

Наприклад, клас польоту асоціюється з класом літака двонаправлено. Асоціація представляє статичне відношення, яке ділиться між об'єктами двох класів.

Агрегація є варіантом взаємозв'язку "має"; агрегація є більш конкретною, ніж асоціація. Це асоціація, яка представляє частково цілі або часткові стосунки. Як показано на зображенні, професор "має" клас для викладання. Як тип асоціації, агрегація може бути названа та мати ті самі прикраси, що і асоціація. Однак агрегація не може включати більше двох класів; це має бути бінарна асоціація. Крім того, навряд чи існує різниця між агрегаціями та асоціаціями під час реалізації, і діаграма може взагалі пропустити відносини агрегування [7].

Агрегація може відбуватися, коли клас є колекцією або контейнером інших класів, але вміщені класи не мають сильної залежності життєвого циклу від контейнера. Вміст контейнера все ще існує, коли контейнер знищений.

В UML він графічно представлений у вигляді порожнистої форми ромба на класі одним рядком, що зв’язує його із наповненням класом. Сукупність - це семантично розширений об’єкт, який у багатьох операціях трактується як одиниця, хоча фізично він складається з декількох менших об’єктів.

Графічне представлення UML узагальнення - це форма порожнистого трикутника на кінці суперкласу рядка (або дерева рядків), що зв’язує його з одним або кількома підтипами.

Відносини узагальнення також відомі як спадкування або відносини "є".

Суперклас (базовий клас) у відносинах узагальнення також відомий як "батьківський", суперклас, базовий клас або базовий тип.

Підтип у відносинах спеціалізації також відомий як "дочірній", підклас, похідний клас, похідний тип, клас успадкування або тип успадкування.

Узагальнення може бути показано лише на діаграмах класів та на діаграмах використання.

При моделюванні UML взаємозв'язок реалізації - це взаємозв'язок між двома елементами моделі, в яких один елемент моделі (клієнт) реалізує (реалізує або виконує) поведінку, яку вказує інший елемент моделі (постачальник).

Графічне представлення UML реалізації - це порожниста форма трикутника на кінці інтерфейсу штрихової лінії (або дерева рядків), яка з'єднує її з одним або кількома реалізаторами. Проста головка стрілки використовується на кінці інтерфейсу штрихової лінії, що з'єднує її з користувачами. У діаграмах компонентів використовується графічна умова «м'яч і сокет» (реалізатори виставляють кульку або льодяник, тоді як користувачі показують сокет). Реалізації можна показати лише на діаграмах класів або компонентів. Реалізація - це взаємозв'язок між класами, інтерфейсами, компонентами та пакетами, що з'єднує елемент клієнта з елементом постачальника. Зв'язок реалізації між класами / компонентами та інтерфейсами показує, що клас / компонент реалізує операції, пропоновані інтерфейсом.

Залежність - це слабша форма зв'язку, яка вказує на те, що один клас залежить від іншого, оскільки він використовує його в певний момент часу. Один клас залежить від іншого, якщо незалежний клас є змінною параметра або локальною змінною методу залежного класу. Це відрізняється від асоціації, де атрибут залежного класу є екземпляром незалежного класу. Іноді відносини між двома класами дуже слабкі. Вони взагалі не реалізовані зі змінними-членами. Швидше вони можуть бути реалізовані як аргументи функції-члена.

Представлення UML асоціації - це лінія, що з'єднує два пов'язані класи. На кожному кінці рядка є додаткові позначення. Наприклад, ми можемо вказати, використовуючи наконечник стрілки, що загострений кінець видно з хвоста стрілки. Ми можемо вказати власність шляхом розміщення кульки, ролі, яку відіграють елементи цього кінця, вказавши ім'я ролі та множинність екземплярів

цієї сутності (діапазон кількості об'єктів, які беруть участь в асоціації з точки зору іншого кінця).

Класи сутності моделюють довгоживучу інформацію, якою обробляє система, а іноді і поведінку, пов'язану з цією інформацією. Їх не слід ідентифікувати як таблиці баз даних чи інших сховищ даних.

Вони намальовані як кола з короткою лінією, прикріпленою до нижньої частини кола. Як варіант, їх можна намалювати як звичайні класи із позначенням стереотипу «сутність» над назвою класу.

Діаграма класів програмного продукту зображена на рисунку 3.2.

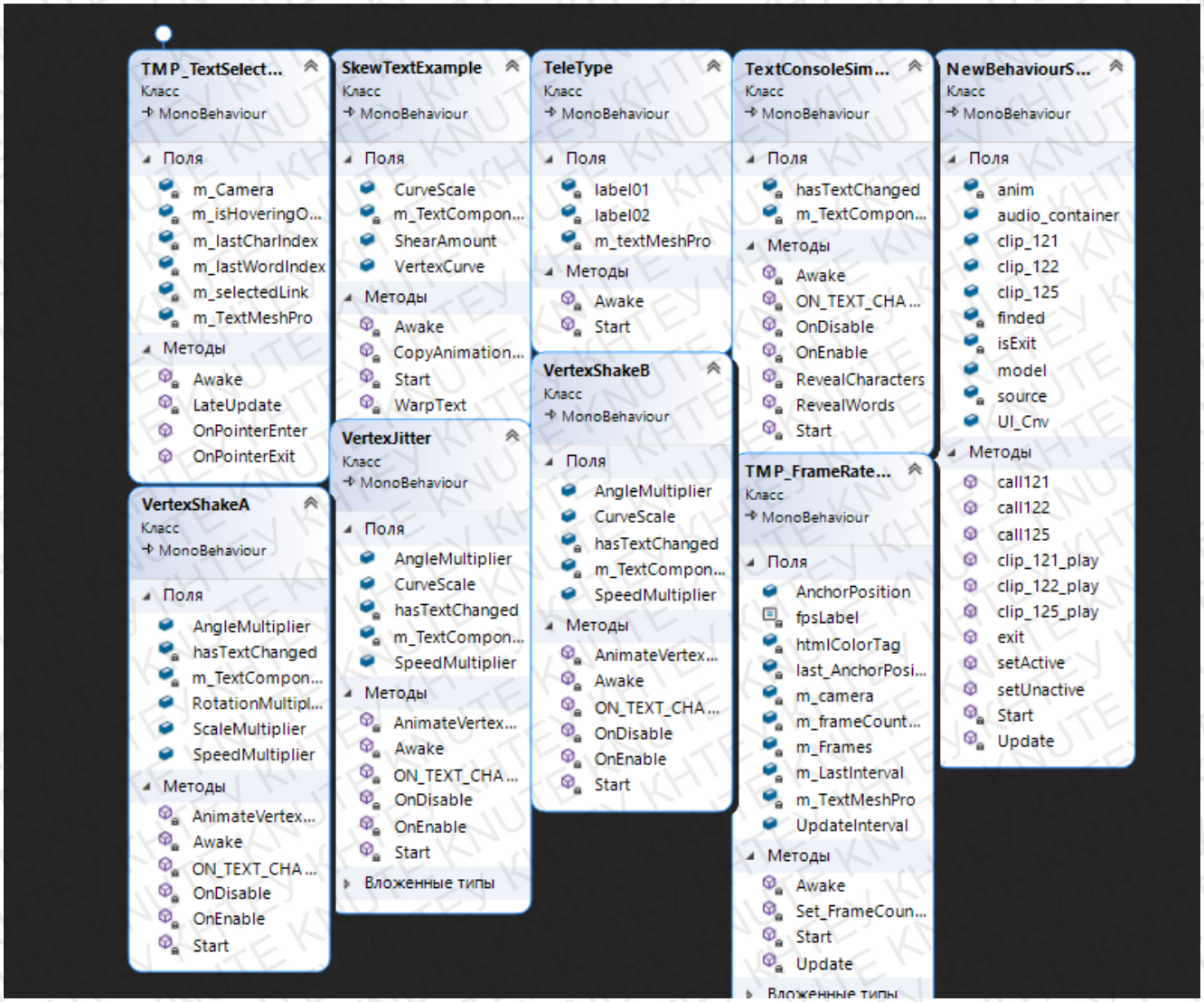


Рис. 0.2 Діаграма класів

3.3. Моделювання процесів

На рисунку 3.3 показано модель основного процесу, а саме модель процесу показу моделі. Дана діаграма показує основні фактори, які впливають на проходження процесу показу моделі користувачеві інтерактивного мобільного додатку доповненої реальності інформаційного спрямування.

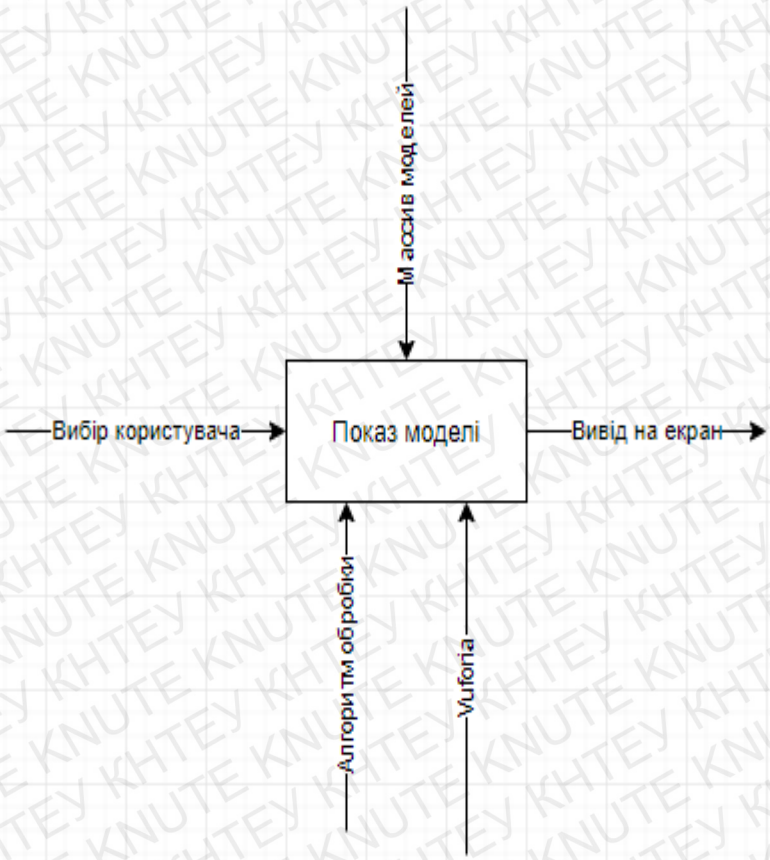


Рис. 0.3 Модель процесу показу моделі в нотації IDEF0

3.4 Висновки до розділу 3

В даному розділі проведено аналіз варіантів використання та проектування моделі додатку доповненої реальності.

На базі інформації, отриманої в результаті аналізу, було сформовано діаграма варіантів використання, діаграма класів та модель процесу.

Вся отримана інформація є основою для подальшої розробки інтерактивного мобільного додатку з використанням технологій доповненої реальності.

На основі третього розділу можна зробити висновок про структуру та поведінку додатку в певних ситуаціях, що відповідають меті та практичній цінності проекту.

РОЗДІЛ 4
РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІНТЕРАКТИВНОГО МОБІЛЬНОГО ДОДАТКУ ДОПОВНЕНОЇ РЕАЛЬНОСТІ ІНФОРМАЦІЙНОГО СПРЯМУВАННЯ

4.1. Налаштування платформи Vuforia

В першу чергу необхідно створити ліцензійний ключ додатку та базу даних маркерів, які будуть служити для розпізнавання.

Спочатку слід створити ліцензійний ключ, для цього використовуємо офіційний сайт сайт Vuforia.

Для створення ліцензійного ключа використовується вкладка Develop -> License Manager. Ліцензійний ключ необхідний для підтвердження наявності ліцензії в пакеті Vuforia всередині Unity, на основі якої пакет формує дозволені види діяльності і доступний функціонал пакету в цілому. При відсутності ліцензійного ключа пакет вважає, що ліцензія відсутня, а отже не доступні навіть найпростіші дії на кшталт відстеження зображення.

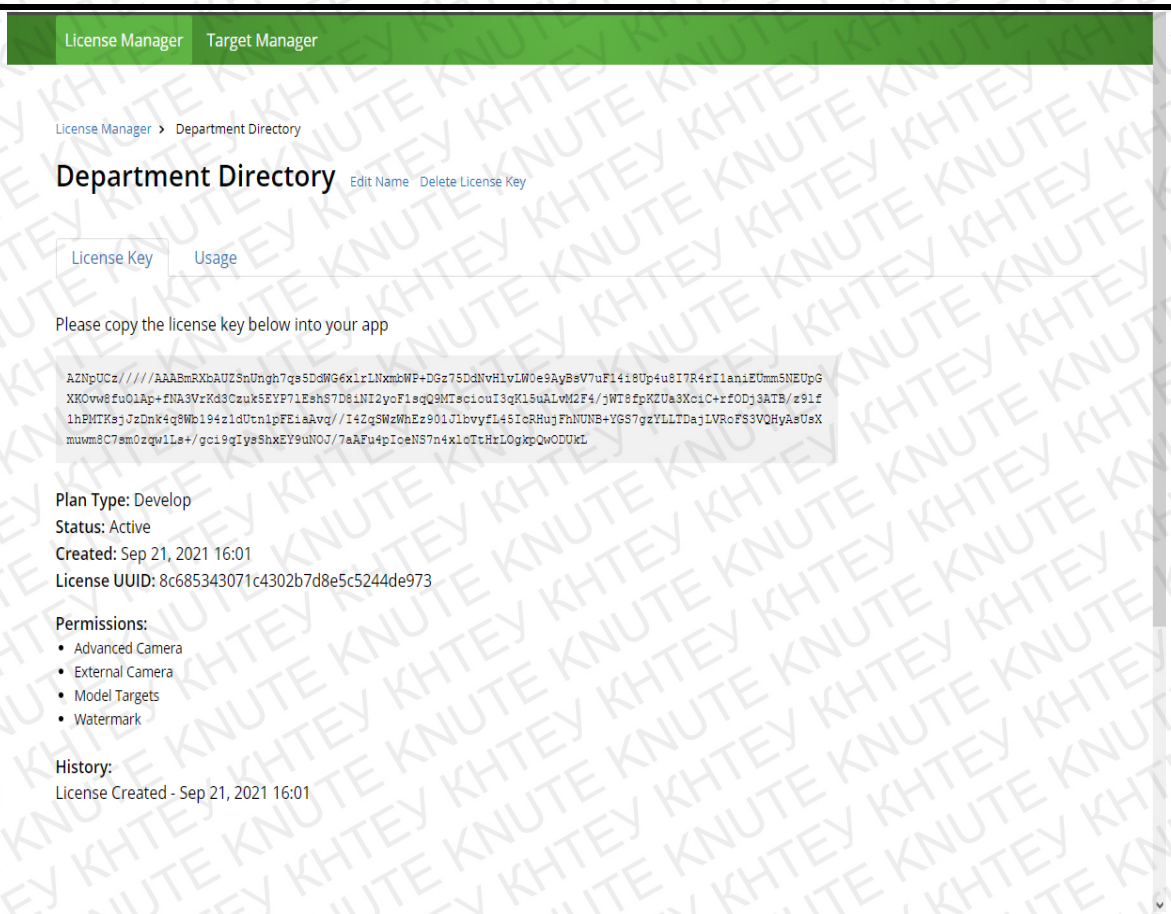


Рис. 0.1 Ліцензійний ключ Vuforia

Після створення ліцензійного ключа необхідно створити так званий «таргет», тобто маркер (рис. 4.2), на який буде орієнтуватися додаток.

Маркер представляє собою логотип кафедри, на якому в подальшому будуть встановлені ключові точки, за допомогою яких буде проводитися розпізнавання. Дане зображення було обрано через велику кількість дрібних деталей і ліній, кожна з яких теоретично може перетворитися на ключову точку. Чим більша деталізація маркера – тим вищий буде коефіцієнт «аугментізації», тобто розпізнаваності маркеру платформою.



Рис. 0.2 Маркер для розпізнавання

На рисунку 4.3. показано вкладку таргетів на офіційному ресурсі Vuforia. Даний таргет має максимальний коефіцієнт аугментизації (п’ять зірок з п’яти), що означає швидке і просте розпізнавання маркеру платформою в подальшому.

Чим вищий коефіцієнт – тим вища вірогідність швидкого розпізнавання таргету всередині кінцевого додатку.



Рис. 0.3 Створення таргету

Після створення усіх необхідних компонентів слід відповідним чином створити сцену в Unity і встановити відповідні налаштування.

Для початку на сцену поміщується AR-камера, Image Target та модель.

Конфігурація Vuforia вимагає, щоб компонент відображуваної моделі був дочірнім для Image Target, тобто в ієрархії знаходився «всередині» Image Target.

Такі вимоги пов’язані з особливостями реалізації механізмів Image Target всередині платформи. Механізм наступний – коли камера розпізнає Image Target всередині нього «вмикається» модель, тобто стає видимою для користувача виключно поверх Image Target.

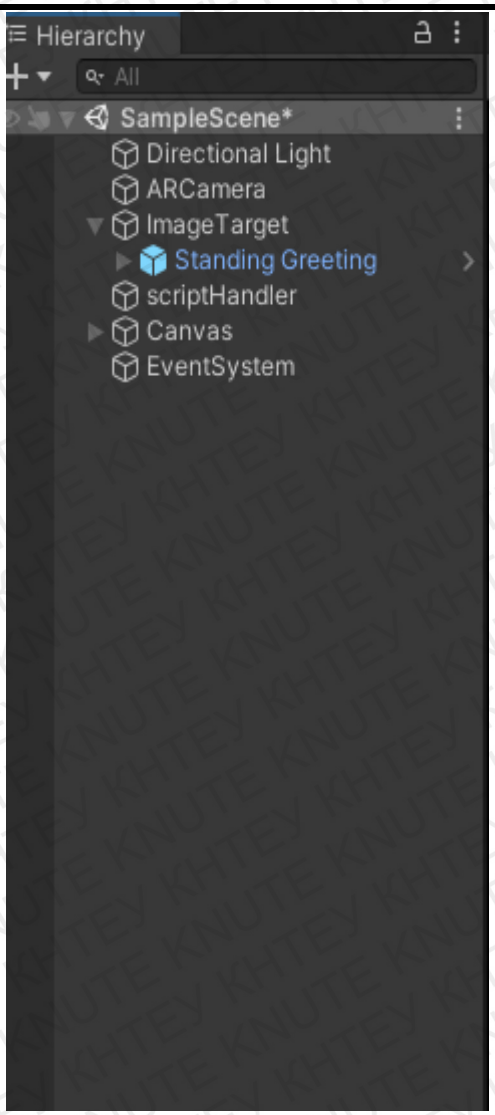


Рис. 0.4 Структура сцени

Далі слід налаштувати камеру та таргет. Фінальний вигляд налаштувань показано на рисунках нижче. Налаштування конфігурації платформи через AR-Camera представляє собою введення ліцензійного ключа, отриманого раніше на офіційному ресурсі Vuforia та виборі бази даних, яка містить маркер .

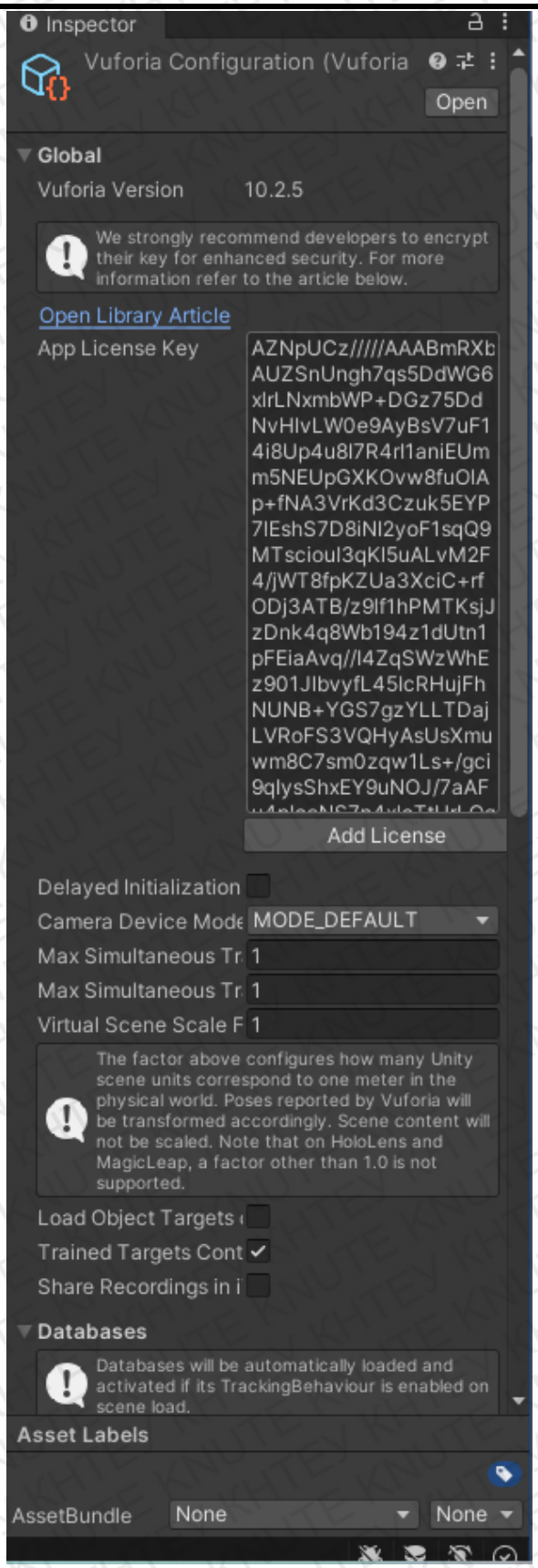


Рис. 0.5 Налаштування камери

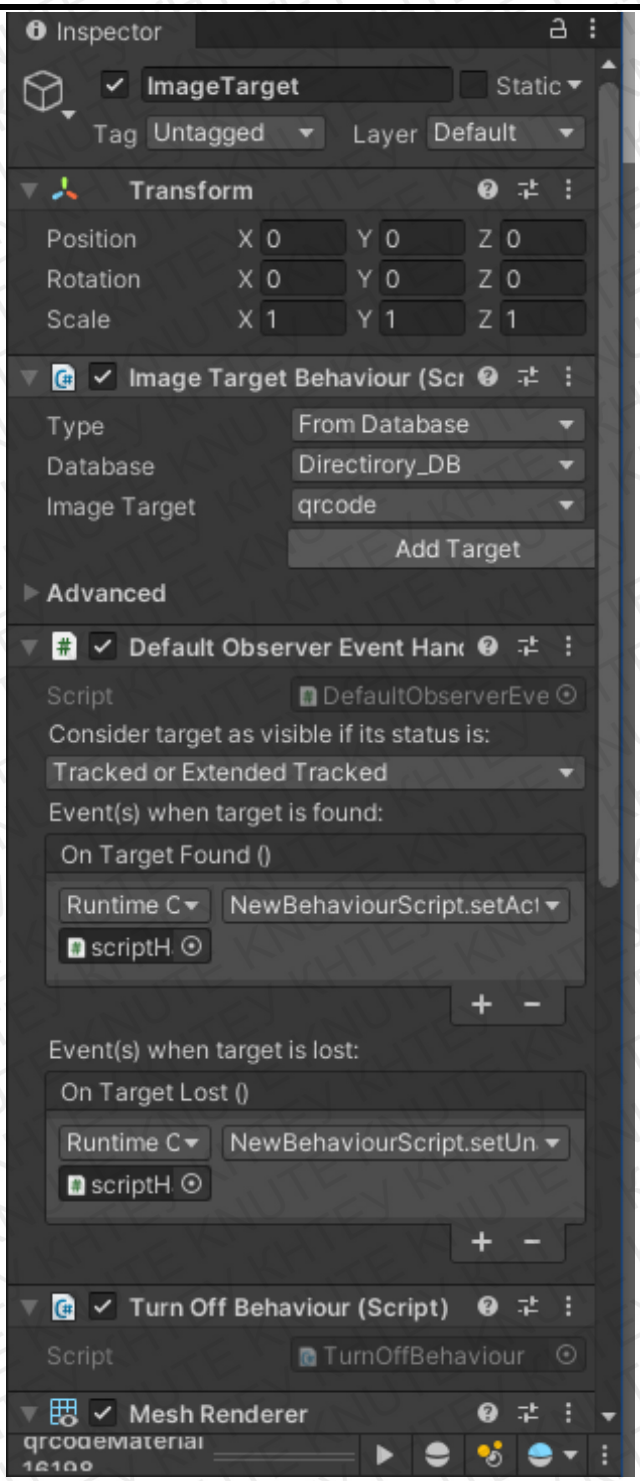


Рис. 0.6 Налаштування таргету

Налаштування таргету полягає в встановленні бази даних таргетів, яка була скачана з офіційного ресурсу після додавання в неї Image Target. Далі з обраної бази даних слід обрати необхідний Image Target, в нашому випадку він один, проте можливості платформи дозволяють додавати безліч таргетів і розміщувати їх на

сцені одночасно, що означає розпізнавання їх всіх без необхідності перезавантаження додатку або завантаження іншої сцени.

4.2. Початок роботи з додатком

На початку роботи з додатком користувачеві не доступне головне меню, воно стає доступним тільки після розпізнавання маркера і вітальних анімацій моделі.

Робота додатку починається із запуску камери і очікування ініціалізації маркеру (рис. 4.7)



Рис. 0.7 Початок роботи

4.3. Відстежування маркера і привітання

Після відстежування маркеру програється анімація привітання (рис. 4.8) і анімація спокою, яка зациклюється і програється доки користувач не натисне будь яку кнопку в меню (рис. 4.9).



Рис. 0.8 Анімація привітання

Описані анімації створені за допомогою технології «кісткової анімації». Тобто, всередині 3Д моделі створюються так звані «кості», які імітують реальні кості людини, а всі рухи базуються на правильній перестановці даних кісток покадрово.

4.4. Робота з програмою

Після анімації модель залишається в позиції спокою, а користувачеві стають доступні кнопки головного меню (рис. 4.9), які дають доступ до інформації про освітні програми і виходу з програми.



Рис. 0.9 Анімація спокою

При натисканні на кнопки з назвами освітніх програм роботи вмикає анімацію розмови (рис. 4.10) і промовляє заготований текст про освітні програми відповідних напрямків. Тексти зберігаються в внутрішніх змінних програми, для кожного тексту збережено окрему змінну з текстом, який з'являється у лівій частині екрану і змінну, в якій зберігається посилання на звуковий файл, в якому збережено даний текст в форматі голосу. Усі перемикання анімацій реалізовані за допомогою компоненту Animator, який є стандартним компонентом Unity і відповідає за набори анімацій окремих моделей і переходи між ними. В компонентів Animator існує поняття тригерів, які стають умовами для запуску переходів між анімаціями. Тригер представляє собою бінарне поле, яке може приймати значення 1 або 0 (True або False) і в залежності від його значення запускається та чи інша анімація. Значення тригеру встановлюється програмно із програмного коду основного управляючого скріпта.



Рис. 4.10 Анімація розмови

Анімація розмови представляє собою повторення типових рухів руками і головою під час розмови і вироблено за допомогою технології кісткової анімації.

4.5 Висновки до розділу 4

В даному розділі проведено аналіз розробки та використання програми.

На базі інформації, отриманої в результаті аналізу, було продемонстровано основні етапи розробки та використання додатку.

Було приділено більше уваги на AR частину розробки додатку та на демонстрацію роботи основних частин програми, які відображають.

На основі четвертого розділу можна зробити висновок про додаток та його відповідності завданням дослідження.

Отриманий результат після реалізації мобільного додатку може використовуватись в умовах кафедри як портативний інтерактивний довідник. Представлений приклад реалізації демонструє, що мобільний додаток можна використовувати в освітній сфері.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

В процесі виконання проекту було розроблено інтерактивний мобільний додаток доповненої реальності інформаційного спрямування, який представляє собою довідник кафедри з використанням технологій доповненої реальності.

Розроблений додаток повністю відповідає заявленому і запланованому функціоналу, налагоджений та готовий до використання в реальних умовах

Як ідею для розвитку можна розглянути додавання більшої кількості моделей помічників і переробку інтерфейсу додатку на більш «приємний для ока». Такі зміни приведуть до більшого бажання користувачів використовувати подібного роду програму.

Також, в процесі розробки додатку було отримано нові і закріплено старі знання щодо розробки додатків в середовищі Unity, програмування мовою C# та роботи з технологією доповненої реальності Vuforia Augmented Reality.

					КНТЕУ 121– 02-03.МР		
					Модель інтерактивного мобільного додатку доповненої реальності інформативного спрямування	*Стадія*	*Аркуш*	*Аркушів*
Зм	*Аркуш*	*№ докум*	*Підпис*	*Дата*		ВП	50	51
Зав. каф.		Криворучко О.В.		01.11.21				
Керівник		Жирова Т.О.		01.11.21				
Гарант		Токар В.В.		01.11.21	*Висновки та пропозиції*	Факультет інформаційних технологій, 2м курс, 2 група		
Розроб		Галич І.В.		01.11.21				

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. C# Language Specification (PDF) (4th ed.). Ecma International. June 2006. Retrieved January 26, 2012.
2. "Visual Studio 2019 Preview Release Notes". docs.microsoft.com. Retrieved October 15, 2020.
3. "Download .NET 5.0 (Linux, macOS, and Windows)". Microsoft. Retrieved October 15, 2020.
4. Axon, Samuel (September 27, 2016). "Unity at 10: For better–or worse–game development has never been easier". Ars Technica
5. McWhertor, Michael (October 22, 2014). "Former EA CEO John Riccitiello is now head of Unity". Polygon. Archived from the original on December 5, 2018. Retrieved October 17, 2018.
6. Smykill, Jeff (August 9, 2006). "Apple Design Award winners announced". ArsTechnica. Archived from the original on January 19, 2015. Retrieved January 18, 2015.
7. Brodkin, Jon (June 3, 2013). "How Unity3D Became a Game-Development Beast". Dice Insights. Archived from the original on October 19, 2018. Retrieved October 30, 2018.
8. Cohen, Peter (October 11, 2007). "Unity 2.0 game engine now available". PCWorld. Archived from the original on March 20, 2019. Retrieved January 14, 2019.

ДОДАТКИ

Додаток А

Лістинг програмного коду

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

public class NewBehaviourScript : MonoBehaviour
{

    // Start is called before the first frame update
    public GameObject model;
    public GameObject UI_Cnv;
    public GameObject audio_container;
    public AudioClip clip_121;
    public AudioClip clip_122;
    public AudioClip clip_125;
    bool finded;
    bool isExit = false;

    private Animator anim;
    private AudioSource source;

    void Start()
    {
        anim = model.GetComponent<Animator>();
        source = audio_container.GetComponent<AudioSource>();
```

```
    }

    // Update is called once per frame
    void Update()
    {
        if(anim.GetCurrentAnimatorStateInfo(0).IsName("sitting idle") && finded)
        {
            UI_Cnv.SetActive(true);
        }
        if(isExit = true &&
anim.GetCurrentAnimatorStateInfo(0).IsName("standing_idle"))
        {
            Debug.Log("Exit");
            Application.Quit();
        }
    }

    public void setActive()
    {
        anim.SetBool("isDetected", true);
        finded = true;
    }

    public void setUnactive()
    {
        anim.SetBool("isDetected", false);
        finded = false;
        UI_Cnv.SetActive(false);
```

```
}
public void call121()
{
    StartCoroutine("clip_121_play");
}
public IEnumerator clip_121_play()
{
    source.clip = clip_121;
    anim.SetBool("talking", true);
    source.Play();
    yield return new WaitForSeconds(source.clip.length);
    source.Stop();
    anim.SetBool("talking", false);
}
public void call122()
{
    StartCoroutine("clip_122_play");
}
public IEnumerator clip_122_play()
{
    source.clip = clip_122;
    anim.SetBool("talking", true);
    source.Play();
    yield return new WaitForSeconds(source.clip.length);
    source.Stop();
    anim.SetBool("talking", false);
}
public void call125()
{
```

```
        StartCoroutine("clip_125_play");
    }
    public IEnumerator clip_125_play()
    {
        source.clip = clip_125;
        anim.SetBool("talking", true);
        source.Play();
        yield return new WaitForSeconds(source.clip.length);
        source.Stop();
        anim.SetBool("talking", false);
    }

    public void exit()
    {
        isExit = true;
        anim.SetBool("good_bye", true);
    }
}
```