Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра інформатики

Курсова робота

освітній ступінь – бакалавр

на тему: «**Розробка інтуїтивного інтерфейсу користувача для доповненої реальності**»

Виконав: студент 3-го року навчання,

Спеціальності

121 «Інженерія Програмного Забезпечення»

Студент Мєдвєдєв Дмитро Романович

Керівник Франків О.О.

магістр комп’ютерних наук, асистент

«16» травня 2021 р.

Київ – 2021

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра інформатики

Освітній ступінь бакалавр

Спеціальність 121 «Інженерія Програмного Забезпечення»

Освітня програма бакалавр

ЗАТВЕРДЖУЮ

Завідувач кафедри інформатики

Гороховський С. С.

“10” жовтня 2020 року

ЗАВДАННЯ

ДЛЯ КУРСОВОЇ РОБОТИ СТУДЕНТУ

Мєдвєдєву Дмитру Романовичу

1. Тема роботи **«Розробка інтуїтивного інтерфейсу користувача для доповненої реальності»**, керівник роботи Франків Олександр Олександрович, магістр комп’ютерних наук, асистент

2. Строк подання студентом роботи 17 травня 2021

3. План роботи

   Анотація

   Вступ

ГРАФІК ПІДГОТОВКИ КУРСОВОЇ РОБОТИ ДО ЗАХИСТУ

№ з/п	ПЕРЕЛІК РОБІТ	Термін виконання	Дата ознайомлення наукового керівника	Підпис наукового керівника	Примітки
1.	Вибір теми, затвердження її на засіданні кафедри та закріплення наукового керівника Узгодження календарного графіка підготовки кваліфікаційної роботи. Ознайомлення студента з критеріями оцінювання кваліфікаційної роботи (п. 8.5).	10 жовтня 2020			
2.	Вивчення джерел літератури, матеріалів архівів, періодичних видань, збір та узагальнення фактів, даних	10 жовтеня 2020 – 20 листопада 2020			
3.	Складання плану каліф. роботи та узгодження з науковим керівником	20 листопада 2020			
4.	Написання розділів роботи	20 листопада 2020 – 20 березня 2021			
5.	Проміжний контроль виконання роботи	01 березня 2021			
6.	Написання кваліфікаційної роботи в цілому, ознайомлення з її першим варіантом наукового керівника	11 січня 2021 – 10 квітня 2021			
	Розділ 1 (постановка проблеми, теоретичні основи, огляд літературних джерел)	01 лютого 2021			
	Розділ 2 (аналітично-дослідницька частина)	01 березня 2021			
	Розділ 3 (проектно-рекомендаційна частина)	01 квітня 2021			
7.	Повне завершення написання кваліфікаційної роботи, оформлення її згідно з вимогами й подання на відгук науковому керівнику	01 квітня 2021 – 16 травня 2021			
8.	Подання кваліфікаційної роботи для перевірки письмових робіт студентів НаУКМА на відповідність вимогам академічної доброчесності,	17 травня 2021			
9.	Публічний захист кваліфікаційної роботи перед екзаменаційною комісією	згідно з розкладом роботи ЕК			

ЗМІСТ

АНОТАЦІЯ

В роботі описано особливості розпізнавання жестів та рук, розглянуто існуючі фреймворки, що вирішують дану задачу. Також розглянуто основи Unity та принципи роботи Unity AR Foundation. Результатом роботи є створення фреймворку, що дозволяє спростити розробку застосунків доповненої реальності з використанням сторонніх інструментів для розпізнавання рук та жестів.

ВСТУП

За останнє століття людство зробило технологічний стрибок уперед. Ще десять років тому телефони були просто засобом для розмов, а зараз є багатофункціональними інструментами, що допомагають спростити наші життя.

З кожним роком все більш актуальною є технологія доповненої реальності. Вона - наступний крок технологічного розвитку. Компанії вкладають мільярди в цей напрям, а експерти прогнозують шалений ріст наступні роки [1].

Однак доповнена реальність створила багато викликів, які досі потребують вирішення. Один з них - взаємодія віртуального з реальним. Зараз взаємодія з пристроями відбувається через сенсорні екрани та інші пристрої вводу, але з розвитком машинного навчання, відкрились нові можливості взаємодій, що зараз спостерігаються у розробках розумних окулярів.

Зважаючи на це, темою даної роботи є розробка інтуїтивного інтерфейсу користувача для доповненої реальності, а саме фреймворку, що може спростити створення AR застосунків, що реагують на жести користувача. Метою даної роботи є огляд та використання існуючих технологій у сфері розпізнавання та AR. Ідеєю є демонстрація можливостей поєднання доповненої реальності з машинним навчанням.

Відповідно можна виділити такі основні задачі: розпізнавання рук та жестів, взаємодія з об’єктами доповненої реальності з їх допомогою та поєднання цих елементів у фреймворк. Таким чином, робота містить три розділи.

В першому розділі розглянуто основні концепти машинного навчання та особливості розпізнавання рук та жестів, наведені приклади існуючих фреймворків.

В другому розділі розглянуті концепції доповненої реальності та можливості AR на базі Unity AR Foundation, описані основи Unity, необхідні для розуміння цієї платформи та практичної частини.

Третій розділ присвячено створенню нового фреймворку на базі вищезгаданих технологій, що спрощує роботу з детекторами рук та жестів в Unity AR Foundation. Також в розділі описано створення гри на основі реалізованого фреймворку та можливості удосконалення розробленого рішення.

Розділ 1. Розпізнавання рук та жестів

1.1. Основні концепти машинного навчання.

Машинне навчання - галузь, що дозволяє вирішувати широкий спектр задач, використовуючи підходи що дозволяють комп’ютеру “навчатися”. Її розділяють на три основні типи: навчання з вчителем, навчання без вчителя та навчання з підкріпленням. Ці основні типи також комбінують та розділяють, утворюючи нові типи, наприклад, напівавтоматичне навчання, що є сумішшю навчання з вчителем та без нього.

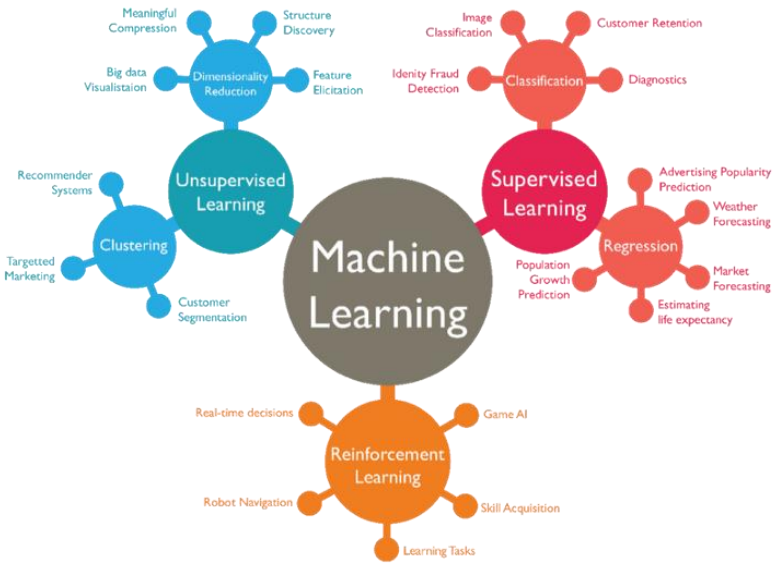


Рис 1.1 Типи машинного навчання і проблеми які вони розв'язують

Однією з ключових складових машинного навчання будь-якого типу є робота з датасетами - набором даних, що може мати різний формат, наприклад текст, фото, відео, аудіо, тощо. Різні методики потребують різних форматів датасетів.

Навчання з вчителем широко використовується для задач класифікації та регресії, тобто коли відомий результат для певного набору даних і потрібно його

отримати для інших даних. Для цього потрібен розмічений датасет - датасет, що містить інформацію про результат. Наприклад, якщо задача полягає в передбаченні ціни квартири за параметрами, треба для кожного набору таких параметрів мати бажаний результат, до якого модель буде прагнути.

Навчання без вчителя використовується для задач кластеризації та зменшення розмірностей, або простіше - у випадках, коли нічого не відомо про дані і розмітка датасету є неможливою. В таких задачах машина шукає схожості між екземплярами даних і таким чином розрізняє їх.

Навчання з підкріпленням найбільше відрізняється від інших. Тут головне не датасет, а середовище в якому перебуває машина і як вона з ним взаємодіє. Unity, наприклад, має власний пакет ML Agents, який використовує такий підхід для тренування спеціального мозку, що може під'єднатися до майже будь-якого об’єкта на ігровій сцені. Основна концепція цього типу - надати машині середовище для взаємодії, на яке вона має вплив, наприклад, ходить, стрибає, і за цієї дії отримує очки. Тобто грає, як і справжня людина. Наприклад, коли машина при смерті отримує мінус 1000 очок, а за кожен крок плюс 1 очко, вона буде вчитися робити максимальну кількість кроків, щоб отримати найбільше очок в результаті. Таким чином, навчання з підкріпленням емулює справжнє навчання людини, але його перевага в тому, що можна створити тисячі середовищ і вчити модель одночасно в кожному з них.

Модель машинного навчання - результат, отриманий при навчанні певного алгоритма. Для кожної проблеми можна підібрати свій алгоритм. Багато з них основані на уявленні щодо даних та математично обґрунтовані. Наприклад, лінійна регресія працює коли є лінійне відношення між даними, KMeans використовують для кластеризації, коли дані розподілені групами, тощо. Але ці алгоритми підходять далеко не для всіх задач. В таких випадках звертаються до нейронних мереж.

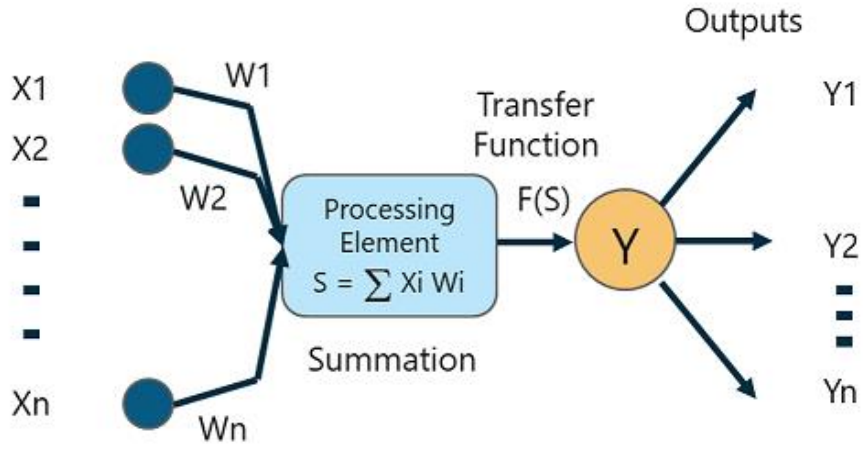


Рис. 1.2 Штучний нейрон

Нейронні мережі - структури, що емулюють частину мозку людини. Вони складаються з нейронів, які формують шари нейронної мережі. Кожен нейрон шару містить параметри, які називають вагами, в кількості рівній кількості входів до цього нейрону плюс один для зміщення. На виході нейрон дає одне значення, що обраховується як сума всіх входів помножених на відповідні параметри та плюс зміщення. Для того, щоб результат не набував будь-яких значень, до нього застосовують функцію, яку називають функцією активації. Прикладом такої функції є ReLU, що нівелює негативні значення, обнуляючи їх. Таким чином, кожен шар нейронної мережі виконує певні трансформації з даними, які використовує останній шар, обраховуючи, наприклад, ймовірності класифікації. Таких шарів може бути багато, вони можуть бути з’єднані як послідовно так і паралельно, що надає нейронним мережам властивість гнучкості. Для аналізу фотографій та відео використовують спеціальні шари нейронних мереж, які називають згортковими. Дані шари будуть детальніше розглянуті в наступній частині.

1.2. Особливості розпізнавання рук та жестів.

Дослідження у сфері розпізнавання рук та жестів зараз набувають популярності, не в останню чергу завдяки бурхливому розвитку технологій доповненої та віртуальної реальності, де активно це застосовується. Розглянемо два підходи, з якими можна підійти до даної проблеми.

Суть першого підходу полягає в тому, що відомо, що потрібно розпізнавати тільки певні жести та рухи руки. Тобто рука та її позиція в кожен момент часу не потрібна, а потрібно знати який жест користувач демонструє зараз. В цьому випадку маємо задачу з розпізнавання певного набору позицій руки, що є задачею класифікації наданих зображень.

Щоб розв’язати таку проблему можна поглянути на підхід до задачі класифікації простих зображень. Перший етап - датасет. Він складається з великої кількості картинок, які розділені по категоріям та покривають всі можливі випадки класифікації. В нашому випадку - створюємо датасет потрібних нам жестів, або використовуємо вже створений. Зараз можна знайти багато датасетів на будь-який смак, від зображень цифр пальцями, до мови жестів. Але тут слід розуміти, що більшість знайдених варіантів просто не підійдуть до вашої проблеми, тому є великий шанс, що доведеться витратити певний час на збір та обробку інформації власноруч. Другий етап - створення та тренування моделі. Тут все залежить від проблеми та сформованого датасету. Зазвичай такі задачі розв’язують використовуючи згорткову нейронну мережу, яка надає можливість машині розкладати зображення на компоненти та надавати їм ваги як це робимо ми при аналізі зображення. Цікаво зазначити, що до розвитку нейронних мереж спеціалісти власноруч вираховували оптимальні фільтри для такого сорту задач, що займало величезну кількість часу та потребувало значних фінансових вкладень.

Розглянемо роботу згорткового шару нейронної мережі детальніше. Будь-яка картинка - це набір, або матриця чисел. Якщо просто перетворити

зображення на вектор і подати як вхід до простої нейронної мережі, то така мережа не зможе навчитися, оскільки не зможе виділити певної закономірності між даними. Більше того, якщо картинка в RGB форматі та має високу роздільну здатність, відповідно матриця її значень дуже велика, це уповільнить навчання моделі в рази. Згорткова мережа вирішує ці проблеми.

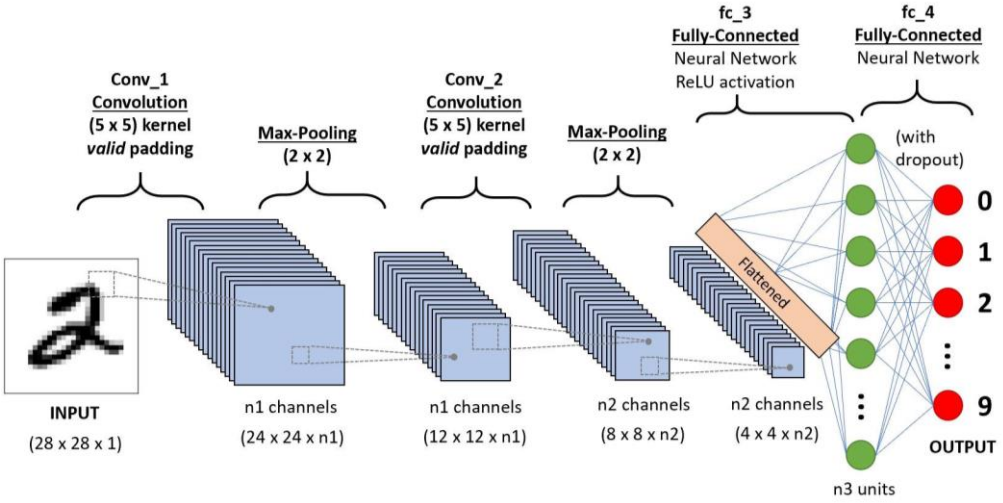


Рис 1.3 Базова згорткова нейронна мережа

Зазвичай застосовують два різні шари: Pooling та Convolution. Вони застосовують підхід накладання маски на зображення. Маска - матриця вказаної розмірності, що складається з вагів і відповідно теж навчається. Процес операції Convolution полягає в тому, що маска накладається на ділянку картинки, перемножує і сумує значення пікселів з маскою і додає зміщення. Таким чином знаходиться новий піксель картинки. Далі з певним кроком фільтр рухається далі по картинці, обраховуючи значення нових пікселів. Операція Pooling - буває або Max-Pooling, де береться найбільший піксель серед накладання, або Average-Pooling, де рахується середнє значення пікселів. Обидві операції дозволяють зменшити розмірність картинки та виділити ознаки картинки, наприклад, краї об’єктів на картинці, частинки цих об’єктів, тощо. Далі, можна подати знайдені ознаки на вхід як вектор до звичайної нейронної мережі.

Отже, отримавши готову модель, потрібно переконатися що вона працює коректно в необхідному нам середовищі. Наприклад, якщо система розробляється для телефону, то необхідно подумати про продуктивність та ефективність, оскільки користувач не буде довго чекати на результат обробки. Також варто пам’ятати про обмеження смартфонів, зокрема про ємність аккумулятора та потужність пристрою.

Загалом, такий підхід дозволяє зосередити увагу на проблемі та вирішити її. З мінусів, тут варто зазначити, що у разі розширення функціоналу, або його зміни, потрібно буде починати все майже з нуля.

Розглянемо альтернативний варіант. Суть цього підходу полягає в тому, що машина працює безпосередньо з рукою. І, знаючи її розташування, робить припущення щодо жесту. Складність цього підходу різна. Можна розпізнавати як базову інформацію про руку - ліва чи права, сторона долоні, так складнішу - знаходити межі руки, або повну інформацію про пальці та всю структуру долоні. Варто розуміти, що розробка такого роду системи може зайняти не один рік і є раціональним використання готових рішень, яких на ринку стає все більше і більше. Розглянемо декілька таких рішень в наступному розділі.

1.3. Огляд існуючих фреймворків

Існує багато фреймворків, які дозволяють інтегрувати функціонал розпізнавання жестів та рук. Розглянемо два з них: MediaPipe та Manomotion.

MediaPipe - відкритий крос-платформенний фреймворк, розроблений компанією Google, що містить багато готових рішень для обробки відео, які використовують підходи та методи машинного навчання. Найбільшою перевагою даного фреймворку є те, що він написаний на мові C++ та може бути легко розгорнутий на будь-якій платформі та операційній системі. Фреймворк

використовує відеокарту та багатопотоковість для прискорення обчислень, тому навіть на мобільних пристроях швидкість його роботи на високому рівні.

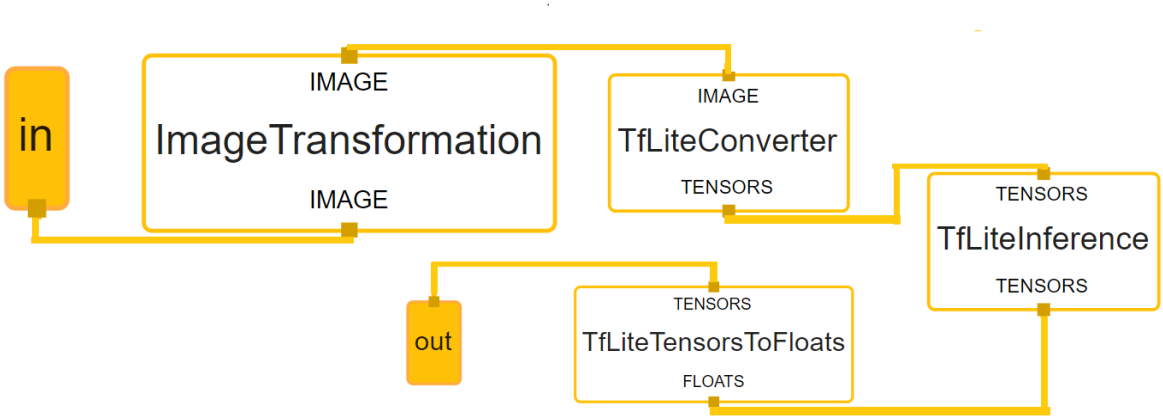


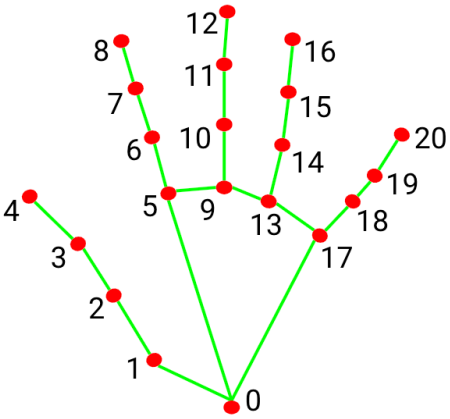
Рис 1.4 Граф для класифікації зображень

Загалом MediaPipe працює на модульній основі. Це означає, що користувач може будувати свої структури з потрібними йому функціями, поєднуючи вже готові компоненти, пропоновані самим фреймворком. Наприклад, проста класифікація зображень задається у вигляді графу, де використовується вже готова нейронна мережа (TfLineInference) з трансформацією вхідних даних для цієї мережі (ImageTransformation, TfLiteConverter) та з перетвореннями результатів роботи мережі (TFLiteTensorsToFloats).

Таким чином отримуємо систему, яка приймає вхідні дані, перетворює їх до формату вхідних даних моделей та трансформує результат обробки до потрібних значень.

Фреймворк пропонує багато готових структур для використання, однією з яких є розпізнавання рук. Дане рішення приймає на вхід фотографію певного розміру та повертає всю інформацію про руку - кількість рук, розташування руки на екрані, 21 ключову точку.

Спочатку використовується модель, що знаходить руку на екрані, після цього обрізається зображення, щоб воно містило тільки частинку з рукою, та подається на вхід моделі, що займається пошуком ключових точок.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Рис. 1.5 Ключові точки руки

MediaPipe цікаве рішення, але є один масштабний недолік, це брак документації та інструкцій по роботі з ним. Однак варто зазначити, що фреймворк досі у стадії активної розробки, яка дає надію на покращення якості документації в майбутньому.

Розглянемо інший фреймворк, який має схожий функціонал, але все ж відрізняється від вищезгаданого - Manomotion. Дане рішення сконцентроване лише на розпізнаванні жестів та рук. На відміну від MediaPipe, доступ до повного функціоналу даного рішення є платним, а сам фреймворк працює в основному з Unity AR Foundation, що і стало головним аргументом для вибору цього застосунку в практичній частині роботи. Безкоштовна версія містить функціонал знаходження розташування руки, деяких її ключових елементів, наприклад, центра долоні, інформацію про стан руки, тобто про жест, та деяку додаткову інформацію. Фреймворк орієнтований на мобільні пристрої, тому показує хорошу швидкість розпізнавання на смартфонах.

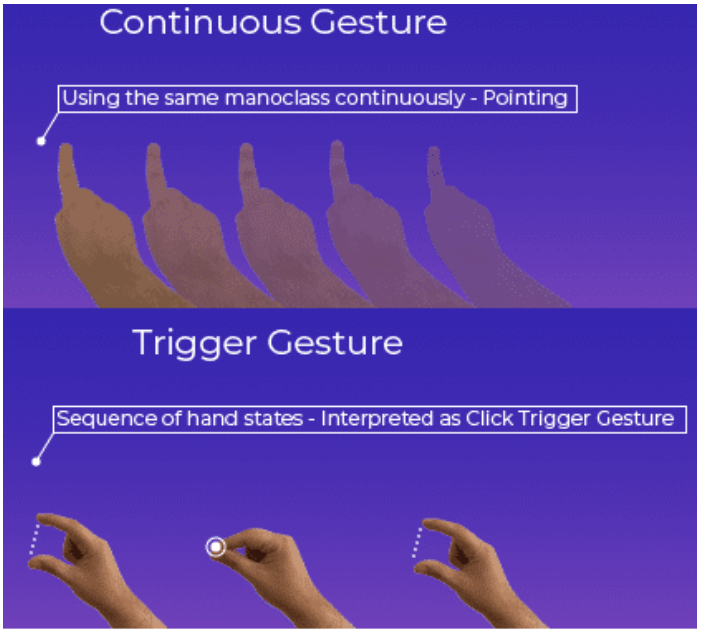


Рис 1.6 Типи жестів в Manomotion

Головною відмінністю від MediaPipe є повернення як результат не тільки ключових точок, а й інформації про жест, які бувають двох типів. Перший тип - TriggerGestures. Це жести які представлені певною послідовністю позицій руки. Наприклад, клік - це швидкий дотик великого та вказівного пальців. Другий тип - Continuous Gestures. Відбувається якщо постійно показувати один тип жесту. Наприклад, взяти та перенести щось в інше місце - це жест Hold, тобто імітація тримання. Інший приклад - це вказування на об’єкт вказівним пальцем. Отже, Manomotion містить низку вбудованих жестів, які можуть комбінуватися в різні послідовності, які в свою чергу можна використати для формування нового жесту.

Якщо розглядати детальніше роботу фреймворку, то жести поділяють на три базові класи: Pinch, Grab, Pointer. Кожен клас містить декілька жестів та стани руки, які позначаються номерами від нуля до шістнадцяти. Ця міра показує наскільки рука, що показує певний жест, закрита, або відкрита. Таким чином, можна ще точніше контролювати реагування на жести та створювати більш гнучкі зв’язки між ними.

Серед недоліків даного рішення варто зазначити неможливість роботи без інтернету, оскільки для роботи потрібно підтвердження ліцензії від сервісу Manomotion. Вона працює за принципом, де до певної кількості завантажень застосунку з фреймворком він безкоштовний, а далі потребує оплати.

1.4. Висновки до розділу

В даному розділі були розглянуті основні концепції машинного навчання, наведені різні варіанти їх використання у вирішенні проблем розпізнавання рук та жестів та описані два фреймворки - MediaPipe та Manomotion, кожен з яких має свої особливості та унікальний функціонал.

Розділ 2. Доповнена реальність на базі Unity AR Foundation

2.1. Основні концепти доповненої реальності.

Розширена реальність (XR) - відноситься до всіх видів реальностей та являє собою загальне поняття згенерованої комп'ютером реальності. Розширена реальність включає в себе доповнену реальність (AR), віртуальну реальність (VR) та змішану реальність (MR). Найбільше відрізняється серед них віртуальна реальність, оскільки вона створює штучне, цифрове середовище, з яким людина може взаємодіяти через спеціальні пристрої, окуляри та сенсори. Доповнена реальність та змішана реальність дуже схожі. Доповнена реальність зосереджується на додаванні до реального світу віртуального контенту. Цей віртуальний контент ніяким чином не взаємодіє з реальним світом, а є лише доповненням до нього. В свою чергу, змішана реальність поєднує в собі найкраще з доповненої реальності та віртуальної реальності, даючи можливість віртуальному світу взаємодіяти з реальним і навпаки.

Загалом AR працює за єдиною схемою. Є камера, що дає зображення світу навколо. Зображення потрапляє в певний обробник, який додає віртуальні елементи до нього, в результаті доповнене зображення з'являється на екрані пристрою. Для того, щоб результуюче зображення правильно наклалось на відео використовуються різні сенсори, наприклад, сенсор глибини, що допомагає зрозуміти відстань між об'єктами, гіроскоп, що слідкує за позицією та кутом нахилу телефона та акселерометр, що слідкує за зміною швидкості, рухом та поворотом пристрою.

Очевидно, що не будь-який пристрій підходить для задач доповненої реальності. Найчастіше використовуються смартфони, оскільки сучасні моделі мають майже все необхідне для роботи з AR. Єдиним недоліком є те, що ця технологія тільки виходить на ринок. Тому, списки пристроїв, що задовільняють

всім потребам технології, невеликий. Можливості сильно залежать безпосередньо від SDK та операційної системи смартфона. Також великі корпорації активно розробляють окуляри, що надаватимуть змогу взаємодіяти з доповненою реальністю, а найбільш амбітні говорять про лінзи, що зможуть замінити окуляри в майбутньому. Крім смартфонів, доповнена реальність використовується на комп'ютерах та будь-яких інших екранах. Яскравим прикладом такого застосування є трансляції різних подій та матчів, де на основне відео накладається віртуальний контент. Існує декілька підходів для роботи з доповненою реальністю. Розглянемо їх детальніше.

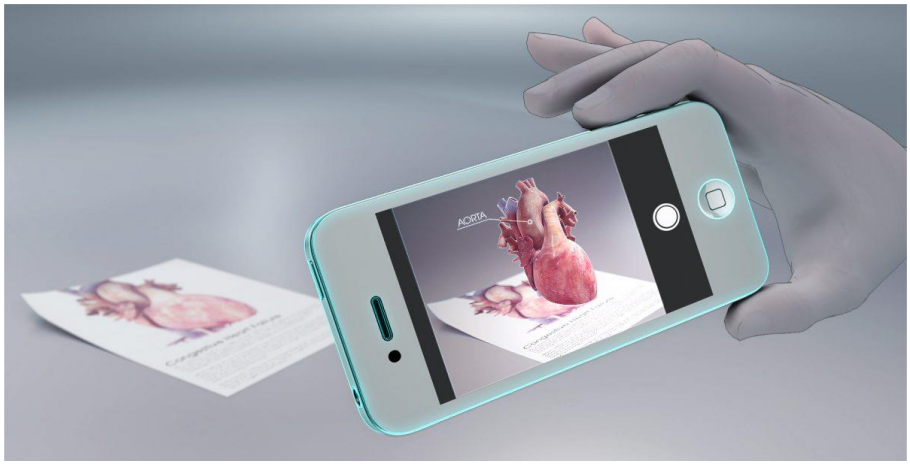


Рис 2.1 Маркерна доповнена реальність

Маркерна доповнена реальність - орієнтована на пошук спеціальних маркерів. Якщо такий маркер знайдено, то відображується запрограмований контент. Самі маркери можуть бути будь-якого типу - QR-код, штрих-код, загалом, будь-яке зображення, яке можна розпізнати. Тому, деколи цей тип називають просто "розпізнаванням зображень". Один з багатьох прикладів - це малюнки, що слугують маркерами і при наведенні вмикають певну анімацію. Найбільшим плюсом такого підходу є те, що потрібно лише знаходити потрібний маркер на фото. Тому не потрібно мати купу сенсорів та постійно слідкувати за

середовищем навколо. Даний тип доповненої реальності працює майже безпомилково і є доступним для широкого спектру користувачів.

Безмаркерна доповнена реальність - не прив’язана до певної локації чи маркеру. Контент може виводитись в будь-якому місці за бажанням розробника, або за бажанням користувача. Прикладом такого типу слугує застосунок від IKEA, який надає змогу додавати віртуальні меблі до кімнати, де знаходиться користувач. Очевидно, що тут велику роль грає орієнтація в просторі. Застосунок повинен мати достатньо інформації, щоб визначити, де розташувати об’єкт та де знаходиться користувач відносно нього. Для цього використовується спеціальна сітка, яка накладається на середовище навколо користувача. Далі вже ця сітка використовується для відображення потрібного контенту. Даний тип доповненої реальності використовує спеціальні алгоритми для орієнтування в просторі, тому є складнішим за маркерний і сильно залежить від пристрою на якому працює.

Рис 2.2 Просторова доповнена реальність

Просторова доповнена реальність - використовує GPS координати замість маркерів. Коли користувач знаходиться в потрібній точці, спрацьовує тригер і ми бачимо контент. Активно використовується компас та гіроскоп, що вбудовані в смартфони. Наприклад, можна створити простий застосунок, що створює

віртуального Санту з маршрутом, що задається точками на карті. Якщо пристрій знаходиться там, де Санта пролітає в даний момент часу, то користувач побачить його віртуальну модель над головою.

Проекційна доповнена реальність - проектує світлові проекції на фізичні об’єкти. Простими словами - це голограми. Спеціальні пристрої допомагають взаємодіяти з проекціями шляхом порівняння звичайної проекції з проекцією, що була змінена, наприклад дотиком.

Доповнена реальність що базується на VIO (візульна інерцільна одометрія) - технологія, що дозволяє орієнтуватися в просторі за допомогою сенсорів та камери. Таким чином пристрій створює віртуально 3D модель середовища навколо, може її оновлювати, знаходити відстані між об’єктами. Даний тип доповненої реальності використовують Apple та Google в своїх ARKit та ARCore відповідно.

2.2. Основи Unity

Unity - платформа для розробки ігор та інших застосунків, що дозволяє писати програми майже для будь-якої системи, від браузера до телефону. Платформа має своє власне середовище розробки, що надає розробникам різні можливості. Наприклад, тестувати свої програми без розгортання на цільовому пристрої, використовувати безліч функціональних пакетів та інші корисні речі.

Основною функціональною одиницею в Unity є сцена. Сцена - це середовище для роботи з контентом. Вона містить в собі всі частини застосунку. Наприклад, гра може мати декілька рівнів і кожен рівень є окремою сценою зі своїми об’єктами та налаштуваннями. Для простих застосунків може вистачити однієї сцени, для комплексних, зазвичай, однієї сцени замало. Також можна

створити копію вже створеної сцени і таким чином тестувати різний функціонал та налаштування, не впливаючи на основну.

Складовими сцени є об’єкти. Вони бувають різні: аудіо, ефекти (певні анімації, наприклад горіння), світло, відео, інтерфейс та самі 2D, 3D об’єкти. Кожен об’єкт на сцені складається з компонентів. Вони дають можливість налаштовувати створений об’єкт будь-яким чином. Як вже було зазначено вище, Unity містить велику колекцію пакетів, що у свою чергу містять свої сцени, компоненти та інші елементи, які можна використовувати у своїх застосунках. Наприклад, можна завантажити будь-яку 3D-модель, додати її на сцену і застосувати до неї компоненти фізичного двигуна Unity. Таким чином віртуальна модель буде під дією віртуальних законів фізики, зокрема під силою тяжіння.

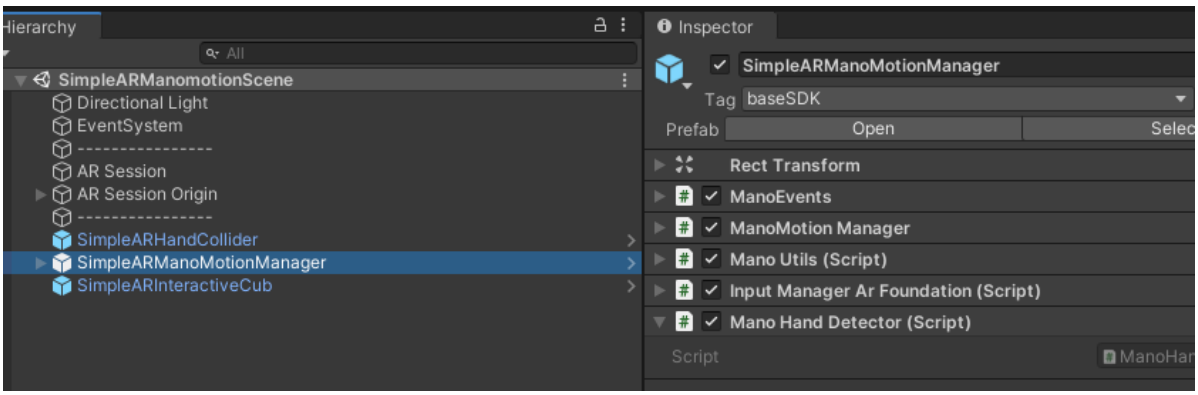


Рис 2.3 Сцена та об’єкт з компонентами

Крім того, що розробник може створити свій власний об’єкт на сцені, він також може створювати власні компоненти. Одними з таких компонентів є скрипти, що написані на мові C#. Ці скрипти надають змогу програмно діяти на об’єкти та реалізовувати власні концепції поведінки. Майже кожна програма на Unity містить в собі скрипт менеджер. Такі скрипти прикріплені до пустих об’єктів, що ніяк не взаємодіють з самою сценою. Прикладом простого менеджера є менеджер подій, що глобально реагує на події натискання різних кнопок і відповідно до події виконує методи, які, наприклад, можуть створювати

нові об’єкти на сцені. Класи скриптів можуть наслідувати спеціальні базові класи, наприклад MonoBehaviour. Клас, що наслідує даний, отримує змогу реалізувати методи життєвого циклу Unity. Таких методів багато, розглянемо лише один - Update, що викликається кожного фрейму. Використання даного методу надає змогу робити дії та перевірки кожного разу коли сцена оновлюється. Якщо гра має 60 кадрів в секунду, то відповідно даний метод спрацює 60 разів. Дані методи необхідно використовувати правильно, оскільки вони мають свої особливості. Наприклад, якщо використовувати фізичний двигун Unity і робити обрахунки сил, що діють на об’єкт, в методі Update, то зміна кількості кадрів в секунду буде впливати на фізичні властивості об’єкта, що не є коректним. Для таких цілей використовують інший метод життєвого циклу FixedUpdate, що має спеціальну сталу частоту виклику.

Важливою частиною Unity є взаємодія між об’єктами на сцені. Для цього платформа має спеціальну систему колізій. Для того, щоб об’єкт став частинкою цієї системи він повинен містити спеціальний компонент під назвою колайдер. Цей компонент надає об’єкту форму взаємодії. Наприклад, для шахової фігури можна поставити колайдер у формі куба, тоді буде існувати шанс, що об’єкт, який не торкнувся насправді фігури, спровокує подію відповідальну за дотик. Для загалу дане поняття більш відоме під назвою Hitboxes. Кожен об’єкт з колайдером може бути тригером - об’єктом, що при взаємодії з колайдером іншого об’єкта спровокує виклик у нього спеціальних методів колізії. Найпростішим прикладом таких методів є OnTriggerEnter та OnTriggerExit, що викликаються відповідно при початку взаємодії тригера з об’єктом та після завершення цієї взаємодії.

В Unity широко використовуються шаблони, або Prefabs - об’єкти, які попередньо створені та налаштовані. Вони вже містять в собі потрібні компоненти для коректної роботи та легко додаються на сцену. Майже всі об’єкти менеджери - це шаблони.

2.3. Unity AR Foundation

Unity AR Foundation - платформа спеціально розроблена для створення AR застосунків на майже всіх підтримуваних пристроях та гарнітурах. Вона працює з ARKit, ARCore, Magic Leap та HoloLens, даючи можливість розробляти застосунок один раз одразу на всі бажані пристрої. Іншою вагомою перевагою даної платформи є велика спільнота користувачів Unity та велика кількість довідкової інформації, зокрема туторіалів та статей.

Functionality	ARCore	ARKit	Magic Leap	HoloLens
Device tracking	✓	✓	✓	✓
Plane tracking	✓	✓	✓	
Point clouds	✓	✓		
Anchors	✓	✓	✓	✓
Light estimation	✓	✓		
Environment probes	✓	✓		
Face tracking	✓	✓		
Meshing			✓	✓
2D Image tracking	✓	✓		
Raycast	✓	✓	✓	
Pass-through video	✓	✓		
Session management	✓	✓	✓	✓

Рис 2.4 Функціонал AR Foundation 2.1

Платформа побудована на базі субсистем, що мають конкретні реалізації для кожного підтримуваного інструмента розробки і називаються провайдерами. Кожна субсистема встановлюється як окремий пакет до основного пакету платформи. AR Foundation не тільки надає спільний інтерфейс для різних інструментів розробки AR застосунків, а й дозволяє використовувати переваги Unity паралельно з функціями доповненої реальності. Оскільки кожна з підтримуваних платформ розвивається окремо, нові функції та можливості в них з’являються не одночасно. Тому AR Foundation містить функціонал, що підтримується тільки деякими інструментами розробки, наприклад, функція

Human Segmentation доступна тільки для ARKit, що означає, що проект з цією функцією на Android працювати не буде.

Щоб почати працювати з AR потрібно створити спеціальний об’єкт (Game Object), який буде містити компонент AR Session. Цей компонент дозволяє керувати життєвим циклом застосунку доповненої реальності, вмикаючи, або вимикаючи функціонал доповненої реальності.

В Unity об’єкти на сцені мають своє розташування. Воно може бути глобальне - відносно початку координат сцени, або відносне - відносно іншого об’єкта на сцені. Так як кожен провайдер працює з системою координат сесії доповненої реальності, для коректної роботи з Unity потрібно перетворювати ці координати та інформацію про об’єкт до значень, що зможе відобразити платформа. Для цього використовується інший об’єкт під назвою AR Session Origin, у якого дочірнім об’єктом є сама AR Camera. Така структура дозволяє рухати камеру та відстежувані об’єкти разом. Об’єкт камери, в свою чергу, містить декілька компонентів: AR Pose Driver - орієнтує об’єкт сцени відповідно до інформації про орієнтацію пристрою, AR Camera Manager - дозволяє налаштовувати камеру, AR Camera Background - додає відеопотік з камери пристрою як фон сцени. Таким чином відбувається синхронізація реального середовища і сцени Unity.

Кожен об’єкт доданий на сцену з правильно налаштованою сесію доповненої реальності буде відображатись як AR об’єкт. Щоб взаємодіяти з такими об’єктами Unity пропонує широкий спектр методів. Один з них - Raycasting. Суть методу полягає в тому, що на сцені створюється промінь, який має основу, напрям та довжину. Якщо промінь перетинає, зустрічає об’єкт сцени, платформа повертає інформацію про цей об’єкт, яку можна використати для реакції. Таким чином, при дотику на екрані пристрою Unity будує віртуальний

промінь, що виходить з точки дотику і має орієнтацію AR камери. Також можна використати систему колізій.

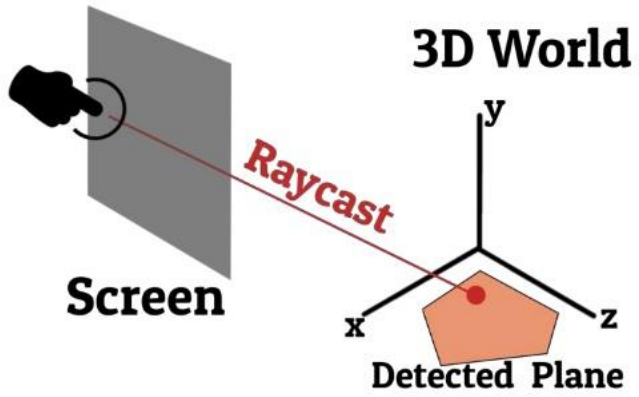


Рис. 2.5 Raycasting

AR Foundation містить також інструменти для розпізнавання картинок та об’єктів, що може бути використано в реалізації маркерної доповненої реальності. Для цього використовуються спеціальні Reference Library, що містять в собі колекції потрібних нам картинок, або об’єктів.

Якщо ж потрібно взаємодіяти з площинами, платформа має функціонал для їх розпізнавання навколо користувача. Кожна розпізнана площина є об’єктом на сцені, тому на ній, або відносно неї можна розміщувати будь-який інший об’єкт. Це є корисним коли, наприклад, потрібно поставити об’єкт на стіл, або іншу площину. Така ж система діє і з розпізнаванням облич. Нові версії AR Foundation також містять функцію occlusion, що дозволяє перевіряти чи є, наприклад, рука користувача перед AR об’єктом, чи вона за нею.

Варто також зазначити, що фреймворк містить велику кількість проблем в роботі, особливо на нових версіях. Зважаючи на це, в практичній частині була використана остання стабільна версія, яка містить набагато менше функціоналу

за найновіші, але не викликала непередбачуваних проблем при розробці та розгортанні на цільовому пристрої.

2.4. Висновки до розділу

В даному розділі були розглянуті основні типи доповненої реальності, основи Unity необхідні для роботи з Unity AR Foundation та сама платформа, яка є містком між найпопулярнішими інструментами розробки застосунків доповненої реальності.

Розділ 3. Розробка фреймворку

3.1. Середовище розробки та передумови

Для практичної частини роботи було вирішено використати AR Foundation, оскільки вона надає змогу розробляти застосунки доповненої реальності під різні платформи, а в нашому конкретному випадку фактично розробляти на операційній системі Windows мобільний застосунок для IOS. ARCore теж має таку можливість, але він більше орієнтований на Android пристрої та має проблеми з роботою на IOS. В свою чергу, AR Foundation пропонує інтерфейс, який використовує інструменти ARKit, що розроблені самою Apple спеціально для IOS.

Рис. 3.1 Unity Game view

Єдиним нюансом залишався сам процес розгортання та тестування застосунку. Для цього потрібно встановлювати віртуальну машину, що працює на MacOS та має можливість скомпілювати створений Unity проект та розгорнути його на мобільному пристрої. Були досліджені всі відомі способи прискорення цього процесу, зокрема компіляція на хмарі, компіляція на

Windows з використанням додаткових інструментів та інші. Найкращим рішенням виявилось використання вбудованої системи тестування в Unity, що надає змогу запускати застосунок без будь-якої компіляції прямо в Unity.

Це працює, коли не потрібна інформація ззовні, а тільки гра. При роботі з AR, застосунок отримує інформацію від пристрою і відповідним чином аналізує. Звичайна Unity на таке не знатна. Рішенням є використання спеціального пакета, який дозволяє підключити пристрій до Unity та використовувати його при розробці застосунку. Для таких цілей Unity створила окремий продукт - Unity Mars та розробляє офіційний плагін, який, відповідно до офіційних форумів, рік тому пройшов стадію тестування, але досі закритий від загалу. Unity Mars та інші користувацькі плагіни, що можуть допомогти, на жаль, платні для всіх, включно зі студентами. Тому, було вирішено створити фреймворк, що дозволить спростити процес розробки AR застосунків та їх тестування в середі Unity.

3.2. Створення фреймворку

Фреймворк було названо SimpleAR. Він складається з декількох папок та скриптів. Для роботи фреймворку потрібно створити об’єкт-менеджер, що буде містити певні компоненти - Input та Detector. Також потрібно мати сторонній детектор жестів та рук. В нашому випадку був використаний Manomotion, але гнучка архітектура фреймворку дозволяє використовувати будь-який інший . Об’єкт менеджер може бути лише один на сцену. Результатом його роботи є виклик події, яка відповідає за конкретний жест. Розгляньмо це детальніше.

AR Foundation надає інформацію про поточний фрейм, з якої можна дістати кадр камери, де застосунок буде знаходити руку. Далі цю картинку відправляємо на відпрацювання сторонньому детектору, що в результаті поверне інформацію про знайдені руки та жести. Це робота компоненти Input. Вона

відповідає за те, щоб фрейм був опрацьований. Іноді компонента має також забезпечити правильний формат даних для сторонніх детекторів. Після того, як результат отриманий, вона тригерить спеціальну подію завершення опрацювання, за якою слідкує інша компонента - детектор. Завдання цієї компоненти перетворити результат, отриманий від опрацювання фрейму, до заданої структури та викликати відповідні події жестів. Компонента детектора повинна розширювати абстрактний клас HandDetector, що містить корисні методи та самі події, що необхідно викликати. Даний клас також містить спеціальний масив, що зберігає останню інформацію про руку. Розмір масиву легко можна змінити на потрібний користувачеві і таким чином отримати фактично історію опрацьованих фреймів, що може бути корисно при створенні власного функціоналу.

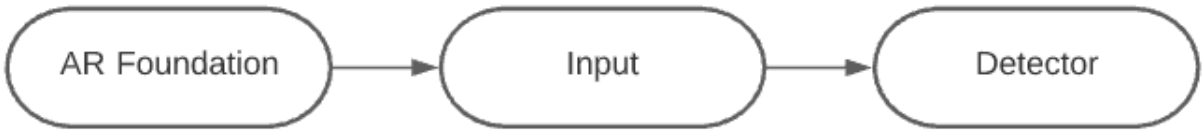


Рис. 3.2 Базова структура фреймворку

Для того, щоб AR об’єкт міг реагувати на вищезгадані події, потрібно додати до цього об’єкту скрипт, що буде наслідувати абстрактний клас ActionBase. Даний клас при створенні підписується на всі події детектора, реалізуючи віртуальні методи, які викликаються при активації конкретної події. Ці методи можна перевизначити, таким чином реалізувавши поведінку об’єкта при жесті, або події.

```
public static Action OnHandDetected;
public static Action OnHandLost;
public static Action OnNoHandAction;
public static Action OnHandClicked;
public static Action OnHandGrabbed;
public static Action OnHandReleased;
public static Action OnHandHold;
public static Action OnHandPoint;
public static Action OnNoHandContAction;
```

Лістинг 3.3 Події детектора

Наразі реалізовано два різних об’єкти-менеджери. Перший використовує Manomotion для розпізнавання рук та жестів, другий - емулює сторонній детектор і використовує кліки та клавіатуру як тригери подій. Використовуючи такий детектор, можна тестувати поведінку застосунку при виявлені жестів в самому Unity, не витрачаючи час на компіляцію та розгортання на пристрої. Ці об’єкти збережені в спеціальні шаблони - Prefabs, тому щоб їх використати потрібно лише додати об’єкт з таким шаблоном на сцену.

Також фреймворк містить шаблон об’єкту, що рухається за рукою. Тобто, кожен фрейм розташування такого об’єкту прирівнюється до координат руки. Це дозволяє використовувати механізми платформи по взаємодії між об‘єктами так, наче користувач реально щось чіпає рукою. Зокрема, використовується механізм колізій, описаний у попередньому розділі. Рука є об’єктом з колайдером, який є тригером, що провокує виклики методів колізії інших AR об’єктів. З іншого боку реалізований шаблон куба, що є об’єктом з колайдером, який чекає на тригер. Якщо об’єкт тригер торкнувся, або всередині куба, то він змінює колір на зелений, а якщо користувач активує подію жеста всередині нього, що є аналогом кліку, то це створить на верхній частині куба маленький кубик. А якщо захоче перевірити жест Hold, тобто тримання, то куб почне обертатися.

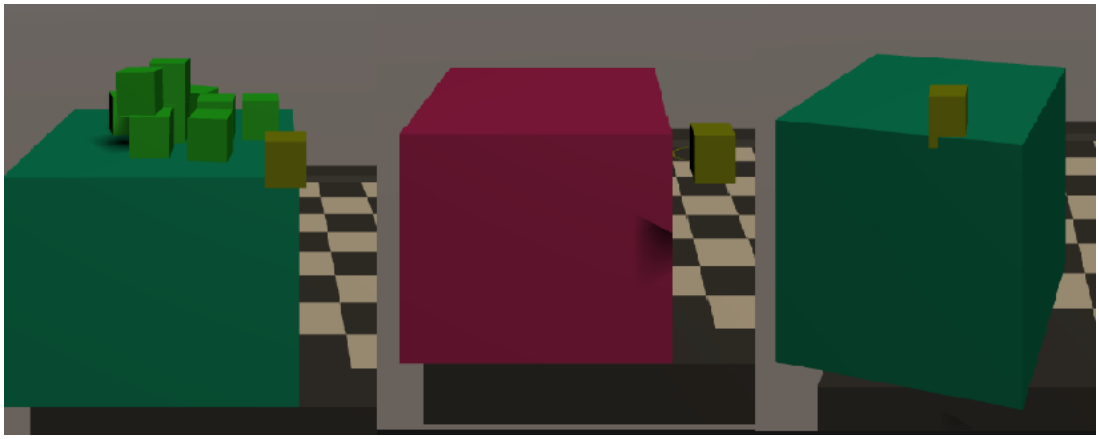


Рис. 3.4 Реакції на події на прикладі куба

3.3. Створення шахів на базі реалізованого фреймворку

Щоб продемонструвати можливості створеного фреймворку було вирішено реалізувати AR шахи, взаємодія з якими буде відбуватися за допомогою рук та жестів. Вони також є частиною фреймворку та знаходяться у папці з прикладами.

Шахи мають дві різні сцени, кожна з яких містить один з об’єктів-менеджерів фреймворка. Перша - для розробки та тестування в Unity, з використанням об’єкта-менеджера (SimpleARMouseManager), що емулює жести та руку. Друга - містить в собі налаштоване AR середовище, яке використовує Manomotion (SimpleARManomotionManager) для відповідно знаходження жестів та рук. Unity надає можливість швидкого пошуку та завантаження будь-яких 3D моделей, тому створення шаблонів шахових фігурок та дошки не зайняло багато часу.

Основною функціональною одиницею в шахах є об’єкт, що містить скрипти Board Manager, Visualization Manager та Board Interaction Manager. Шаблон цього об’єкту також присутній в прикладах.

```
public class BoardInteractionManager : ActionBase
{
    Frequently called  0+1 usages  new *
    protected override void OnHandClicked()
    {
        var cell = boardManager.SelectedCell;
        if (cell.x < 0 || cell.x >= BoardManager.Size.x || cell.y < 0
        var figure = boardManager.SelectedFigure;
        if (figure != null && (figure.Cell.x != cell.x || figure.Cell.
        else boardManager.SelectFigure(cell.x, cell.y);
    }
```

Лістинг 3.5 Використання класу ActionBase в BoardInteractionManager

Board Interaction Manager розширює клас ActionBase, що, як згадувалось вище, надає нам можливість реагувати на події жестів. Таким чином, задача даного скрипта заключається у взаємодії з користувачем. Наприклад, саме тут відбувається моніторинг переміщення руки і відповідно підсвітка потрібної клітинки та реакція на бажання користувача вибрати, перемістити фігуру по дошці.

Visualization Manager відповідає за роботу з об’єктами шаблонами. Він містить в собі посилання на об’єкти-фігури та об’єкти-підсвітки, що відображаються при вибору фігури, або просто коли рука вказує на дошку. Коли потрібно створити модельку фігури та розташувати її в коректну клітинку, то програма звертається до методів цього скрипта.

Board Manager - скрипт, що поєднує вищезгадані. Він є центром гри та має можливість розпочинати і закінчувати гру та містить всі необхідні змінні поточного стану гри. Наприклад, вибрану клітинку, розташування фігур та можливі ходи вибраної фігури.

У грі використовуються різні методи виявлення взаємодії користувача з дошкою. Перший - Raycasting. Можна пускати промінь в заданому напрямі з місця розташування об’єкту-руки, наприклад вниз, тобто якщо рука над дошкою,

то знаходимо відповідну клітинку і взаємодіємо з нею. Альтернативно, можемо використовувати AR Camera об’єкт і центр екрану пристрою як відправну точку променю, а напрям визначати розташуванням руки відносно цього центру. Тоді, буде здаватись що користувач наче вказує рукою на потрібну клітинку. Залишатиметься лише можливість взаємодії з фігурами, наприклад їх підняття та переміщення, використовуючи жести Grab, Hold, Release. Це реалізовано з використанням об’єкта руки, який при взаємодії з фігурою і активації потрібного жеста робить вказані дії. Якщо ж рука не взаємодіє з фігурою, то активація події жеста ігнорується.

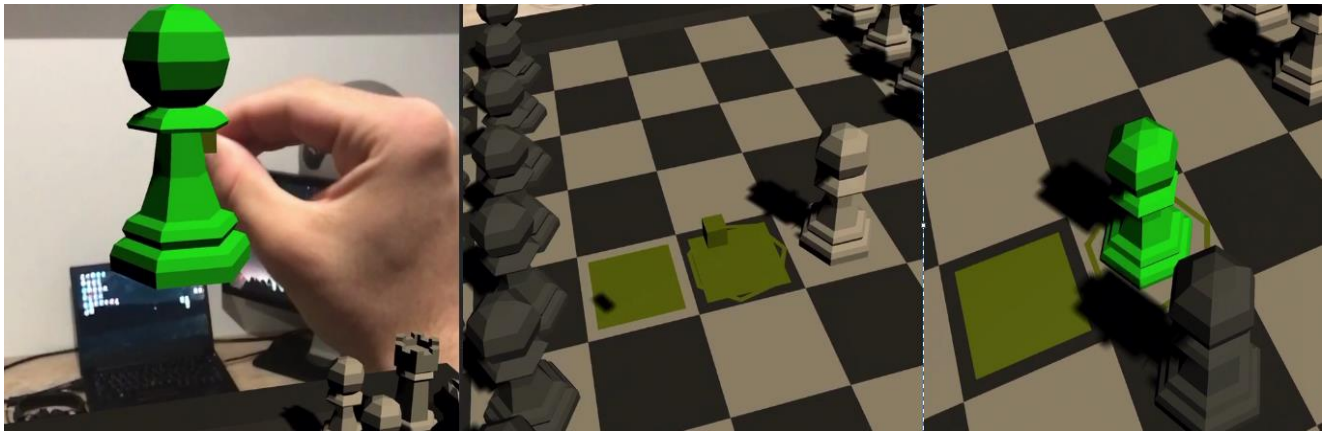


Рис. 3.6 Взаємодія користувача з грою

Загалом процес гри такий. При запуску гри BoardManager створює фігури, збегігаючи інформацію про створені фігури на дошці. Кожен фрейм BoardInteractionManager оновлює інформацію про вибрану клітинку і повідомляє про це BoardManager, який в свою чергу провокує оновлення підсвітки. Коли фреймворк повідомляє про подію кліку, BoardInteractionManager надає інформацію про це BoardManager, який обраховує можливі ходи вибраної фігури. Якщо ж при спрацюванні кліку фігура вже обрана і вибрана клітинка є серед доступних ходів, то відбувається переміщення фігури, якщо потрібно з видаленням фігури що знаходиться на цільовій клітинці. Майже на кожну

вищезгадану дію BoardManager викликає необхідні методи VisualizationManager для оновлення візуалу дошки.

Таким чином фреймворк SimpleAR дозволяє зручно реалізувати скрипт взаємодії користувача з грою та провести більшу частину розробки та тестування гри ні разу не розгорнувши її на цільовому пристрої, що економить дуже багато часу, та надає зручний інтерфейс для взаємодії подій, що дозволяє підвищити читабельність та простоту коду користувача.

3.4. Можливості удосконалення

Приклади в Unity дуже важливі, оскільки демонструють як можна використовувати функціонал та шаблони об’єктів імпортованих пакетів. Тому першим кроком вдосконалення SimpleAR є збільшення кількості прикладів його використання.

Далі, слід згадати Manomotion, який використовується для виявлення рук та жестів. Як було зазначено в огляді даного фреймворку, він не працює без інтернет з’єднання, оскільки потребує перевірки ліцензії, що робить застосунки непрацездатними в ситуаціях і місцях, де немає інтернету. Більше того, Manomotion працює з жестами не ідеально. При тестуванні було виявлено, що іноді жести з’являються довільно, без бажання користувача, що робить їх використання непередбачуваним, оскільки не можна дізнатися коли який жест спрацює і чи справді цей жест виконав користувач. Також новий функціонал AR Foundation, зокрема функція occlusion, не підтримується Manomotion і викликає ситуацію в якій розпізнавання рук фактично припиняється, через помилки в програмі.

Рішенням даних проблем може бути використання іншого фреймворку для розпізнавання, наприклад MediaPipe. Але тут є ряд обмежень, оскільки далеко не

всі такі фреймворки підтримують інтеграцію з Unity, зокрема існує плагін для використання MediaPipe в Unity, але він не є офіційним, а користувацьким і містить достатньо багато проблем при встановленні та використанні. Або можна ігнорувати проблему інтернет з’єднання і перенести всі операції зі знаходженням рук та жестів на сервер, що зменшило б вагу застосунку і відкрило б можливості для використання майже будь-якого фреймворку для розпізнавання.

Іншою можливістю для вдосконалення SimpleAR є збільшення кількості подій, інформації про руку та прикладів використання. Але слід зауважити, що дана інформація напряму залежить від використаного фреймворку для розпізнавання. Тому є доцільним створення базової структури, яку можна буде легко розширити у потрібний користувачеві формат.

Якщо ж говорити про шахи, то тут простір для покращення майже безмежний. Unity надає можливості використання анімацій та ефектів, що додали б яскравості гри. Цікавою можливістю для покращення є реалізація комп’ютерного гравця на основі, наприклад алгоритму Minimax, що покращить можливості гри та надасть додаткового інтересу користувачам. Було б чудово використати можливості AR розташовувати дошку на певній площині, наприклад на столі, та додати можливість збільшення зменшення дошки жестами для зручності користувача. Також створення простого UI додало б грі логіки та сенсу, наприклад, меню дозволило б запускати та зупиняти гру в будь-який момент та змінювати інші налаштування, зокрема складність супротивника та режим гри.

3.5 Висновки до розділу

В даному розділі було описано середу розробки AR Foundation та передумови створення фреймворку. Було розкрито реалізацію даного

фреймворку, описано його структуру та наведено приклад використання. Також було розглянуто реалізацію шахів на базі створеного фреймворку.

ВИСНОВОК

Результатом даної роботи є розгляд існуючих фреймворків для розпізнавання жестів та рук, їх використання у доповненій реальності. Розглянуто базові концепти машинного навчання та підходи з якими можна вирішити проблему розпізнавання. Описані основні концепти доповненої реальності, її типи та особливості. Також розглянуто особливості роботи платформи Unity AR Foundation, зокрема наведено огляд основ Unity для розуміння роботи загалом та описано декілька способів взаємодій між об’єктами доповненої реальності.

Було розроблено фреймворк, використовуючи Manomotion та AR Foundation, описані причини та переваги його розробки, найголовніша з яких полягає в економії часу на компіляцію та розгортання застосунку на цільовому пристрої. Розроблений фреймворк - спроба вирішити дану проблему шляхом використання середовища Unity, для запуску застосунку без будь-яких попередніх дій. Тому, застосування цього рішення однозначно позитивно впливатиме на розробку, тестування та покращення застосунків доповненої реальності, що використовують сторонні фреймворки, або платформи для розпізнавання рук та жестів.

Для демонстрації розробленого фреймворку було створено гру шахи в доповненій реальності, взаємодія з якими відбувається жестами користувача. Наявні і інші приклади використання, що також демонструють можливості поєднання функціоналу Unity з AR середовищем.

Робота містить інформацію про можливі покращення розробленого фреймворку та прикладів до них, які є важливою частиною клієнтських фреймворків в середовищі Unity.

СПИСОК ПОКЛИКАНЬ

1. MGI Notes from the AI frontier Modeling the impact of AI on the world economy [Електронний ресурс] / McKinsey Global Institute. - 2018. - Режим доступу до ресурсу: https://www.mckinsey.com/~/media/McKinsey/Featured%20Insights/Artificial%20Intelligence/Notes%20from%20the%20frontier%20Modeling%20the%20impact%20of%20AI%20on%20the%20world%20economy/MGI-Notes-from-the-AI-frontier-Modeling-the-impact-of-AI-on-the-world-economy-September-2018.ashx
2. Yagang Z. New Advances in Machine Learning / Zhang Yagang., 2010. – 374 с. – (InTech).
3. 14 Different Types of Learning in Machine Learning [Електронний ресурс] / Jason Brownlee. - 2019. - Режим доступу до ресурсу: https://machinelearningmastery.com/types-of-learning-in-machine-learning/
4. Dive into Deep Learning / [Z. Aston, C. Zack, L. Mu та ін.]. // Amazon Science. – 2021. – С. 1025.
5. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way [Електронний ресурс] / Sumit Saha. - 2018. - Режим доступу до ресурсу: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
6. Mediapipe Documentation [Електронний ресурс] / Google Inc. - 2021. - Режим доступу до ресурсу: https://google.github.io/mediapipe/
7. Documentation - Manomotion [Електронний ресурс] / Manomotion Inc. - 2021. - Режим доступу: https://www.manomotion.com/documentation/
8. The Important Difference Between Augmented Reality And Mixed Reality [Електронний ресурс] / Bernard Marr - Режим доступу: https://bernardmarr.com/default.asp?contentID=1912
9. Доповнена реальність (Augmented Reality, AR) [Електронний ресурс] / Lookinar - Режим доступу: https://lookinar.com/uk/rozyasnennya/dopovnena-realnistaugmented-reality-ar/

10. Creating a Location-Based Christmas AR Experience in Less Than 10 Minutes [Електронний ресурс] / Daniel Fortes. - 2018. - Режим доступу: https://medium.com/@daniel.mbfm/creating-a-location-based-christmas-ar-experience-in-less-than-10-minutes-3f900bd35e27
11. Unity Documentation [Електронний ресурс] / Unity Inc. - 2021. - Режим доступу: https://docs.unity3d.com
12. AR Foundation Documentation [Електронний ресурс] / Unity Inc. - 2021. - Режим доступу: https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.1/manual/index.html