

Міністерство освіти і науки України  
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЇВО-МОГИЛЯНСЬКА  
АКАДЕМІЯ»

Кафедра інформатики факультету інформатики



**Сучасні підходи до побудови VR застосунків**  
**Текстова частина до курсової роботи**  
**за спеціальністю «Інженерія програмного забезпечення»- 121**

Керівник курсової роботи  
к.ф.-м.н. Шабінська М. О.  
(прізвище та ініціали)

\_\_\_\_\_  
(підпис)  
“ \_\_\_\_ ” \_\_\_\_\_ 2020 р.  
Виконав студент Рибка М.К.  
(прізвище та ініціали)

\_\_\_\_\_  
(підпис)  
“ \_\_\_\_ ” \_\_\_\_\_ 2020 р.

Міністерство освіти і науки України  
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ

Зав. кафедри інформатики,  
к. ф.-м. н. С. С. Гороховський

\_\_\_\_\_  
(підпис)

“ \_\_\_\_ ” \_\_\_\_\_ 2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ  
на курсову роботу

Студента Рибки Максима Костянтиновича факультету інформатики 3 курсу  
ТЕМА Сучасні підходи до побудови VR застосунків

Вихідні дані:

Зміст ТЧ до курсової роботи:

Календарний план

Вступ

Розділ 1. Існуючі методи та засоби для створення VR застосунку.

Розділ 2. Поетапне створення проекту на різні VR девайси з поясненням.

Висновки

Список використаної літератури

Дата видачі “ \_\_\_\_ ” \_\_\_\_\_ 2020 р. Керівник

\_\_\_\_\_  
(підпис)

Завдання отримав

\_\_\_\_\_  
(підпис)

### Календарний план виконання роботи

#### Тема: Сучасні підходи до побудови VR застосунків

№	Назва етапу	Термін виконання	Примітка
1.	Отримання теми курсової роботи	09.10.2019	
2.	Пошук літератури за темою роботи	10.11.2019	
3.	Ознайомлення з науковою літературою	10.12.2019	
4.	Визначення структури програми	29.02.2020	
5.	Написання першої частини курсової роботи	30.04.2020	
6.	Написання другої частини курсової роботи	05.05.2020	
7.	Написання висновків курсової роботи	09.05.2020	
8.	Перегляд змісту роботи з керівником	-	
9.	Внесення змін до роботи	-	
10.	Створення презентації	10.05.2020	
11.	Завантаження курсової роботи	11.05.2020	

## Зміст

Вступ.....	5
Розділ 1. Існуючі методи та засоби для створення VR застосунку.....	7
1.1 Рушії та різниця між ними.....	7
1.2. Пропоноване програмне забезпечення.....	8
1.3. Mixed Reality.....	10
1.4. Нові технології в MR.....	11
Розділ 2. Поетапне створення проекту на різні VR девайси з поясненням.....	13
2.1. Створення проекту на Unity з використанням VRTK.....	13
2.2. Створення аватару користувача. ІК.....	16
2.3. Hand-tracking.....	20
2.4. Оптимізація застосунку.....	24
Висновки.....	26
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	27

## Вступ

VR застосунки є корисними для представлення проекту у будь-якій галузі, вони допомагають зануритися та здобути досвід, що не можна отримати іншими засобами. Я наведу приклади використання VR у деяких сферах. Віртуальна реальність має величезний потенціал та корисність у сфері навчання. Наприклад, коли необхідно натренувати персонал послідовному виконанню операцій, адже отримати ці навички, працюючи з реальним обладнанням, інколи просто немає можливості, як от при роботі з високою напругою або з обладнанням, що є в обмеженому доступі. В сфері маркетингу різноманітні бренди можуть розширити своє охоплення та популярність створюючи певний “VR Experience”. В сфері медицини віртуальна реальність є незамінною, завдяки 3D візуалізації створюються унікальні проблеми складних операцій, які можна зустріти тільки при ретельному плануванні. Нещодавно VR відіграв важливу роль у розлученні з’єднаних близнюків у масонській дитячій лікарні в Міннеаполісі, визначивши потенційні ускладнення, завдяки детальній моделі тіл близнюків, створеній з УЗД та МРТ сканів [1]. Проте найпопулярнішою сферою застосування VR є розважальна. Віртуальна реальність допомагає користувачу отримати досвід через його відчуття: зір, слух, дотик та інші. Отримані враження від відеоігор, віртуальних турів та екскурсій є набагато яскравішими та запам’ятовуються надовше.

**Актуальність дослідження.** Враховуючи, що віртуальна реальність є новою та постійно розвивається, з’являються нові методи для створення застосунків. Вони вирішують деякі проблеми і недоліки, надають нові можливості та способи використання нових технологій, прискорюють розробку. Створюються засоби для підтримки мультиплатформенності додатку, задля розширення кола користувачів та збільшення доступності. Пристрої нового покоління підтримують нові технології,

такі як “Hand-tracking” для Oculus Quest, що надають більше можливостей та покращують сприйняття, зручність, якість, тощо. Саме тому використання найсучасніших методів для розробки VR застосунку є дуже важливим.

**Об’єкт дослідження** – процес побудови VR застосунку.

**Предмет дослідження** – особливості побудови, використовуючи сучасні методи.

**Мета** – дослідити та розкрити сучасні методи побудови застосунку для VR, проаналізувати їх корисність.

**Завдання дослідження :**

- 1) Дослідження існуючих засобів для побудови застосунку.
- 2) Дослідження нових технологій та методів.
- 3) Аналіз корисності нових методів.
- 4) Поетапне створення VR додатку.

## Розділ 1. Існуючі методи та засоби для створення VR застосунку.

### 1.1 Рушії та різниця між ними.

Перш за все, щоб створити VR застосунок, необхідно визначитися який ігровий рушій використовувати. На сьогодні найпопулярнішими ігровими рушіями, що використовуються для розробки VR є Unity, Unreal Engine, Crytek. Розглянемо яка між ними різниця.

Unity – має найбільшу аудиторію, через це в інтернеті є дуже багато інформації та готових рішень для розробки в цьому рушії. Розробка для VR не є винятком. З'являється все більше і більше розробників, що створюють свої готові рішення для швидкої імплементації необхідного функціоналу. Величезним попитом користується VRTK. Вони надають можливість додати найнеобхідніший функціонал для VR всього за декілька хвилин. В цьому плагіні реалізовані майже всі існуючі способи переміщення у просторі, взаємодії з оточенням та найголовніше – можливість використовувати один проект для створення білда на різні VR девайси, без необхідності завантажувати інші плагіни. Це дуже важливо, адже для кожного з девайсів виробники створюють свої програмні забезпечення для розробки, саме через це з'являються конфлікти в побудові мультиплатформенного додатку. VRTK вирішує цю проблему. Плагін є безкоштовним, як і сама Unity але, використовуючи безкоштовну версію, на початковому екрані завжди буде Unity логотип. Саме через доступність цього ігрового рушія та великої кількості плагінів, поріг входження для Unity є найменшим.

Unreal – також безкоштовний ігровий рушій з винятковою графікою. Як і в Unity існують готові плагіни, але їх набагато менше стосовно VR в порівнянні з Unity, тому в більшості випадків доводиться реалізовувати функціонал самотійно. Частіше за все Unreal використовують для створення проекту для десктопних VR шоломів, тобто для тих, що під'єднуються до

комп'ютера та використовують його потужність, а не для автономних девайсів, що не потребують з'єднання з комп'ютером, адже мають власну операційну систему та необхідне обладнання, тому що Unreal потребує потужне обладнання для отримання гарної графіки. В Unreal VR технологія також може використовуватися для розташування окремих об'єктів в сцені, що є корисним застосунком для левел-дизайнерів.

CryEngine – як і попередні рушії є безкоштовним. Має високий поріг входження нових розробників та набагато слабшу підтримку спільноти. Тому розробка для VR набагато складніша та займає більше часу для досягнення того ж результату, що можна отримати при використанні конкурентів – Unity та Unreal. Але в порівнянні з Unity, використовуючи CryEngine, є можливість досягти більших та кращих результатів, адже на мою думку, він є потужнішим.

Отже, наразі Unity є набагато зручнішим та легшим для використання при створенні VR додатку. Для нього створено більше готових рішень та корисних плагінів. Але його конкуренти мають свої переваги та постійний розвиток в напрямку VR, тож необхідно перевіряти та підбирати найкращий варіант для поставленої задачі.

## 1.2. Пропоноване програмне забезпечення

Для кожного з VR девайсів існує певне SDK(software development kit). Це є необхідною базою для створення VR додатку, адже надається можливість спілкуватися з API певного девайсу та передавати додатку важливі дані, взяті з його датчиків. Але необхідно розуміти, що для кожного SDK можливим є тільки створення додатку для конкретних пристроїв. Наприклад, при використанні Oculus SDK можливо створити тільки додаток для Oculus Go, Gear VR, Oculus Rift, Oculus Rift S та Oculus Quest. Розглянемо особливості при використанні деяких з них.

Oculus SDK – насправді має декілька різних варіантів для Unity, Unreal, Native Windows, Native Mobile та перелік корисних інструментів:

- Oculus ADB Drivers – для встановлення з'єднання з пристроєм, що використовує операційну систему Android;
- Cubemap Viewer – для перегляду кубічних текстур, що використовуються як 360° зображення для перегляду на сторінці додатку після публікації в Oculus Store;
- Oculus Mixed Reality Capture Tools, що дозволяє переносити ваше зображення у віртуальний світ за допомогою веб-камери, зеленого екрану та програмного забезпечення для трансляцій;
- OVR Metrics Tool – для отримання таких показників, як частота кадрів, тепло, навантаження на центральний процесор та графічний процесор.

Також пропонується функціонал для віддаленого управління пристроєм, що є дуже корисним при розробці на автономні пристрої. Щоб почати використовувати Oculus SDK в Unity, необхідно просто завантажити відповідну версію плагіну Oculus Integration з офіційного сайту Oculus та імпортувати її в проект. В цьому плагіні надаються готові сцени для демонстрації певного функціоналу. На відміну від Unity, в Unreal версії більше за 4.10, підтримка Oculus SDK вже зашита для VR проектів.

SteamVR SDK – надає можливість створення застосунку для пристроїв, що працюють використовуючи програмне забезпечення SteamVR : Oculus Rift, Valve Index, HTC Vive та Lenovo Mixed Reality. В додаток до створених засобів для переміщення та взаємодії з оточенням для віртуальної реальності, сучасний плагін SteamVR для Unity та Unreal має готовими 3д моделі рук, оброблює введення з контролерів, змінює та анімує створену руку. Для цього використовується інверсна кінематика до кожного пальця 3д моделі руки, відповідно до вашого інпуту.

PSVR dev kit – щоб отримати доступ у використанні цього програмного забезпечення, необхідно стати зареєстрованим розробником.

Але існують створені програмні забезпечення, що контролюють взаємодію з кожним SDK та дозволяють створювати мультиплатформенний додаток. Найбільшою популярністю користуються OpenVR, що використовується в Unreal Engine і Unity, та VRTK, створений тільки для використання в Unity.

OpenVR – програмний інтерфейс (API), що дозволяє взаємодіяти з дисплеями віртуальної реальності, не покладаючись на SDK конкретного постачальника. До нього стабільно додається підтримка нових пристроїв та технологій. Рушії можуть використовувати OpenVR для отримання положення та орієнтації будь-яких гарнітур VR та застосовувати їх до основної камери. Потім Unity надсилає зображення камери на OpenVR. Він виконує деякі операції над зображенням, а потім відображає його на реальному екрані гарнітури. Це дозволяє рушіям працювати з більшістю гарнітур VR з коробки, без необхідності встановлення сторонніх бібліотек.

VRTK – це безкоштовний плагін для Unity, який пропонує колекцію рішень для розробки віртуальної реальності, насамперед спрямований на те, щоб розробникам було простіше та швидше створювати застосунок та додавати базовий функціонал. Як і OpenVR, працює використовуючи існуючі SDK різних постачальників. Проте VRTK пропонує рішення до проблем, що часто виникали при побудові VR застосунку. Присутні різні методи для переміщення у просторі такі як телепорт, переміщення контролером або перетаскування оточення відносно себе, взаємодії з оточенням та все необхідне для симуляції VR напряду з Unity, без необхідності створення збірки на пристрій, тощо [3].

### 1.3. Mixed Reality.

MR (Mixed Reality - змішана реальність) об'єднує реальний світ та цифрові елементи. У змішаній реальності ви взаємодієте з фізичним та віртуальними предметами, оточеннями. MR використовує технології

комп'ютерного зору та сканування, створює “point cloud” модель оточення та обробляє її. Для використання змішаної реальності пристрій обов'язково має камери на зовнішній стороні, наприклад, у нових пристроїв від Oculus, Rift S та Quest. В старих версіях використовувалися спеціальні датчики, що відстежують групи ІК-світлодіодів. Саме завдяки камерам та датчикам можливо рухатися у 6 DoF. Альтернативою є 3 DoF, що вважається застарілим. DoF (degrees of freedom – ступінь свободи) – термін для руху навколо осі або вздовж осі. У реальному світі всі об'єкти рухаються саме у 6 різних напрямків, тобто 6 DoF. Стосовно VR, DoF використовується для опису осей, відносно яких відстежується переміщення. Відстеження походить від можливості контролювати зміну кута або відстані на осях за допомогою апаратних засобів. 3 DoF означає відстеження тільки орієнтації. Це означає, що відслідковуються 3 осі, навколо яких об'єкт може обертатися. Використовується у старих автономних гарнітурах, як Oculus Go, Gear VR, Pico Goblin 1 та новий Goblin 2. Ці пристрої здатні відслідковувати зміни кутів при обертанні, але не здатні відслідковувати переміщення у просторі. Інколи цього достатньо, адже багато VR додатків на мобільні пристрої передбачають сидіння на місці та отримання досвіду, подібного до їзди на гірках. 6 DoF дозволяє відслідковувати як і орієнтацію гарнітури так і її положення. Переміщення відслідковується у всі 3 осі. Гарнітури, що використовують 6 DoF дозволяють користувачеві пересуватися, ходити навколо об'єктів, розглядати їх під різними кутами.

#### 1.4. Нові технології в MR

Завдяки змішаній реальності, у VR з'явилися такі технології, як просунута Guardian система для відображення визначених реальних меж ігрової зони. Коли користувач наближається до краю межі, напівпрозора сітка відображається у додатку, а якщо користувач вийде за межі цієї сітки, то камери, що знаходяться на зовнішній стороні гарнітури, почнуть передавати зображення з реального світу. Ігрові межі створюються як point-

cloud данні, та, завдяки цьому, можливим є розпізнавання старих ігрових меж, щоб користувачеві не доводилося при кожному запуску ставити новий Guardian.

Також, нещодавно стало можливо використовувати Hand-tracking у віртуальній реальності. Ця технологія розблоковує нові механіки для розробників та творців у віртуальній реальності. Але наразі цю технологію можливо використовувати тільки на Oculus Quest. Відстеження рук дозволяє користувачам бути більш виразними, отримувати більш глибокий соціальний досвід та занурення у віртуальну реальність. Також, ця технологія зменшить поріг входження у VR для людей, яким некомфортно працювати з контролерами. Ще краще, що руки завжди при вас, не потрібно постійно тримати контролери, перевіряти чи вони зарядженні або з'єднувати їх з гарнітурою[8].

Та оскільки ця технологія знаходиться у розробці, програми, що використовують цю функцію, наразі не приймаються для надсилання у Oculus маркет. Наразі Hand-Tracking можливо використовувати тільки на Oculus Quest та Windows Mixed Reality гарнітурах HoloLens 1, 2. Щодо розробки для Windows Mixed Reality для Unity, існує SDK, що називається Mixed Reality Toolkit (MRTK). Він за своїм призначенням схожий на VRTK, має реалізованими функціонал та інтерфейси, що часто використовується для виклику системних функцій. Завдяки MRTK можливо створювати UWP (Universal Windows Platform) проект та завантажувати додаток на сам пристрій.

## Розділ 2. Поетапне створення проекту на різні VR девайси з поясненням.

### 2.1. Створення проекту на Unity з використанням VRTK.

Перш за все, необхідно створити проект в Unity. При створенні нового проекту є можливість обрати шаблон (Рис. 1), він змінює налаштування на основі загальних найкращих практик.

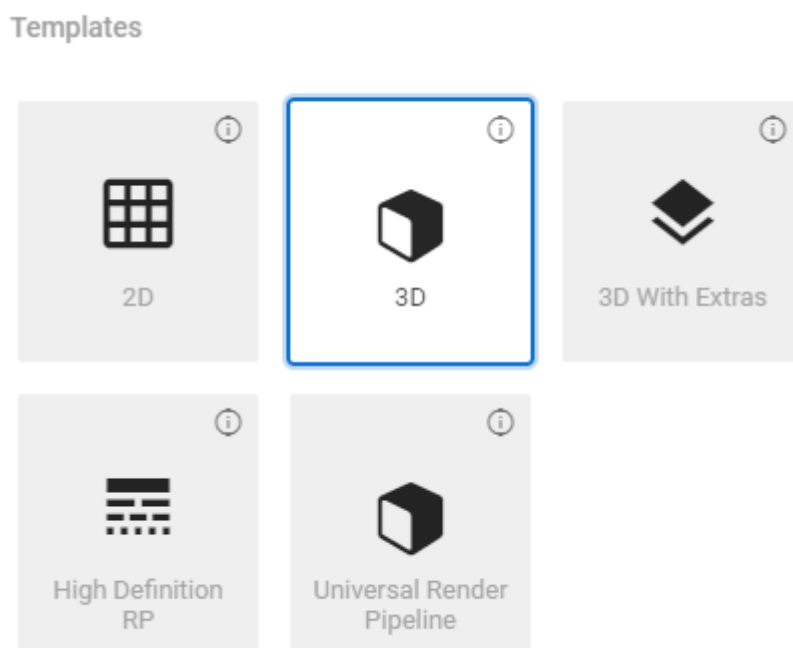


Рис. 1

Шаблони прискорюють процес підготовки проекту та використовують деякі важливі плагіни, що не включені у початковий проект за замовчуванням. Існують такі шаблони:

- 2D – налаштовує проект для розробки 2D-додатку, створює камеру з ортографічною проекцією, налаштовує освітлення та додає плагін для роботи зі спрайтами<sup>1</sup>.

<sup>1</sup> Прості 2D об'єкти, що мають графічне зображення. Unity використовує їх за замовчуванням, знаходячись у 2D режимі.

- 3D – проект за замовчуванням, не додаються плагіни та не змінюються налаштування.
- 3D With Extras – налаштування за замовчуванням, використовуються вбудована функція пост-обробки<sup>2</sup>, також в початковий проект додано набір ассетів<sup>3</sup> з демонстраційним контентом.
- High Definition Render Pipeline – налаштування параметрів проекту для отримання максимальної якості зображення. Включає плагіни Shader Graph, для зручного створення шейдерів, використовуючи візуальний редактор вузлів, та вбудовану пост-обробку. Цей шаблон більше використовують для створення застосунку на комп'ютери, адже потребує високих системних вимог.
- Universal Render Pipeline – налаштування проекту, коли найголовнішим є продуктивність, підтримка всіма платформами та легкість кастомізації графіки. Включає плагіни Shader Graph, та використовує Universal Render Pipeline, що дозволяє створювати оптимізовану графіку на широкому діапазоні платформ.

Для VR найкращим варіантом є останній шаблон – Universal Render Pipeline, адже потрібна підтримка багатьох платформ, як Android та PlayStation. Також, через те, що для комфортного VR досвіду необхідно отримати 60-90 fps<sup>4</sup>, дуже важливою є оптимізація, отже обираємо останній варіант.

Після завантаження та імпорту VRTK (в проекті використовую версію 3.3.0) та всіх необхідних SDK, потрібно додати префаб<sup>5</sup> об'єкту “[VRTK\_SDK Manager]”, саме він забезпечує конфігурацію підтримуваних SDK та керує списком налаштувань для кожного SDK. Разом з усіма можливими SDK, з'являється можливість використовувати симулятор віртуальної реальності. Він дозволяє запускати проект у самому редакторі,

<sup>2</sup> Надає можливість накладати фільтри та ефекти до камери.

<sup>3</sup> Набір моделей, аудіо, зображення або будь-яких файлів, що підтримує Unity

<sup>4</sup> Frames per second – кількість кадрів в секунду

<sup>5</sup> Екземпляр певного об'єкту, зміни якого застосовуються до всіх інших екземплярів

без необхідності збірки проекту, що є дуже корисним та зберігає багато часу, особливо при великих проектах, збірка яких може займати більше двадцяти хвилин.

Для переміщення користувача у просторі в VRTK реалізовані такі методи:

- Basic Teleport – телепортація користувача, вказуючи кінцеве положення завдяки вказівнику. Вказівник може бути прямою лінією або кривою Безьє. Також додається плавне затемнення екрану при переміщенні.
- Height Adjust Teleport – переміщення користувача на різні рівні по у-осі.
- Dash Teleport – плавно переміщає користувача у вказане положення, використовуючи вказівник.
- Переміщення кнопками – переміщення по x та у осі, використовуючи джойстик або тачпад на контролері.
- Drag World – переміщення завдяки перетаскуванню оточення відносно себе.

Щоб використати кожен з цих варіантів, достатньо тільки додати відповідні екземпляри до сцени. Оскільки переміщення у віртуальній реальності намагається обдурити вестибулярний апарат, це викликає ефект запаморочення. Отже найкращим варіантом є Basic Teleport, який у момент зміни положення та оточення затемнить екран.

VRTK передбачає, що розробникам інколи зручно перевизначити існуючу реалізацію для створення певного функціоналу, тож всі класи, що використовуються у екземплярах, є віртуальними.

Наприклад для того, щоб відслідковувати об'єкт з яким взаємодіємо, необхідно перевизначити існуючий клас `VRTK_InteractGrab`, додати

оголошення події `OnObjectGrabbed` та перевизначити метод `PerformGrabAttempt`, для виклику створеної події.

```
public class Custom_VRTK_InteractGrab : VRTK_InteractGrab
{
    public event Action<GameObject> OnObjectGrabbed;
    override protected void PerformGrabAttempt(GameObject obj)
    {
        base.PerformGrabAttempt(obj);
        OnObjectGrabbed?.Invoke(obj);
    }
}
```

Таким чином можна використовувати класи та екземпляри VRTK для реалізації будь-яких задач.

## 2.2. Створення аватару користувача. ІК.

Для додатку віртуальної реальності дуже важливим є максимальне занурення користувача. Щоб досягнути відчуття присутності у віртуальній реальності, важливу роль грає використання моделей рук замість контролерів та створення коректної анімації взаємодії з об'єктами завдяки інверсній кінематиці.

ІК (inverse kinematics, інверсна кінематика) – математичний процес обчислення змінних параметрів гнучких об'єктів для досягнення необхідної позиції [4]. Широко використовується у комп'ютерній анімації та робототехніці, наприклад для кінематичного ланцюга, кінематичної пари або анімації скелету персонажа у грі. Завданням інверсної кінематики є пошук певного набору конфігурацій зчленувань, що забезпечить максимально м'який, точний та швидкий рух до заданої позиції. Використовується для визначення кутів з'єднань та їх положень в моделі, шляхом мінімізації зваженої похибки квадрата, математична формула формулюється так [5]:

$$\sum_{i=1}^n w_i \left( x_i^{\text{exp}} - x_i^{\text{subject}} \right)^2 + \sum_{j=1}^m w_j \left( \theta_j^{\text{exp}} - \theta_j^{\text{subject}} \right)^2$$

Рис. 2

Де  $n$  і  $m$  – кількість маркерів і кутів з'єднання, відповідно.

$x_i^{\text{exp}}$  – позиція експериментального маркера, кінцева позиція.

$x_i^{\text{subject}}$  – положення симуляційної моделі.

$\theta_j^{\text{exp}}$  та  $\theta_j^{\text{subject}}$  – значення  $j$ -го кута з'єднання для експериментальної та симуляційної моделі, відповідно.

$w_i$  та  $w_j$  – значення ваги, що дають можливість змінювати позиції маркерів і кути суглобів по різному. Впливають на те, наскільки слід задовольнити відповідність до кінцевої позиції.

У Unity, щоб налаштувати інверсну кінематику для персонажу, є готові функції у компонента Animator:

SetIKPositionWeight та SetIKRotationWeight, для встановлення ваги цілі інверсної кінематики. Ціль інверсної кінематики – це кінцеве положення та обертання конкретного з'єднання.

SetIKPosition і SetIKRotation – встановлюють позицію та кут обертання цілі інверсної кінематики.

bodyPosition та bodyRotation – положення і обертання центру тіла маси у світовому просторі.

Також в Unity можливо використовувати готові налаштування інверсної кінематики для скелету гуманоїда. Необхідно тільки перейти у налаштування моделі, в розділ Rig, та встановити тип анімації як Humanoid(Рис. 3).

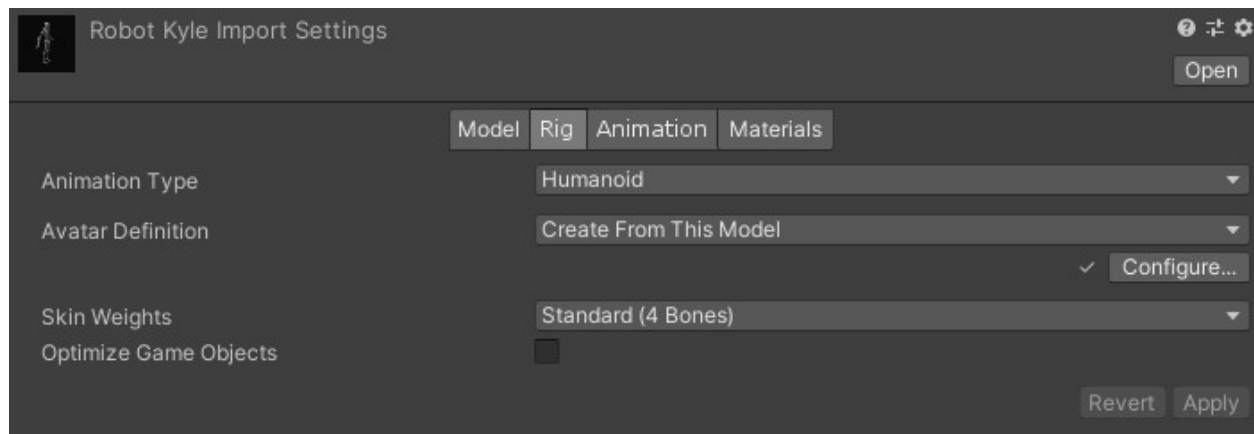


Рис. 3

Це створить аватара для моделі та спробує самостійно виставити налаштування інверсної кінематики, відповідно до скелету гуманоїда.

Наступним кроком потрібно додати компонент Animator та до нього Animator Controller зі створеною анімацією. Неважливо яка саме анімація створена, вона потрібна щоб аніматор викликав OnAnimatorIK кожен фрейм.

Тепер можна створити компонент для виклику функцій інверсної кінематики.

Створений компонент необхідно додати до моделі з компонентом Animator, та написати реалізація методу OnAnimatorIK, що буде оновлювати позиції частин тіла відносно заданих цілей. Оскільки обов'язково, щоб створений компонент знаходився на одному об'єкті з компонентом Animator, можемо додати RequireComponent тег. Він перевіряє наявність вказаного компонента, та при відсутності, автоматичного його додає. Для віртуальної реальності як цілі визначаємо положення контролерів та гарнітури.

```
private void OnAnimatorIK()
{
    /*Перевірка додана, щоб була можливість вимкнути інверсну
    кінематику*/
    if (ikActive)
```

```

    {
        /* Якщо визначена ціль для лівої руки, то викликаємо встановлення
ваги, та позиції цілі.*/
        if (leftHandTarget != null)
        {
            animator.SetIKPositionWeight(AvatarIKGoal.LeftHand, 1);
            animator.SetIKRotationWeight(AvatarIKGoal.LeftHand, 1);
            animator.SetIKPosition(AvatarIKGoal.LeftHand,
leftHandTarget.position);
            animator.SetIKRotation(AvatarIKGoal.LeftHand,
leftHandTarget.rotation);
        }

        if (rightHandTarget != null)
        {
            animator.SetIKPositionWeight(AvatarIKGoal.RightHand, 1);
            animator.SetIKRotationWeight(AvatarIKGoal.RightHand, 1);
            animator.SetIKPosition(AvatarIKGoal.RightHand,
rightHandTarget.position);
            animator.SetIKRotation(AvatarIKGoal.RightHand,
rightHandTarget.rotation);
        }
    }

    /* В іншому випадку встановлюємо ваги позиції та обертання як 0 */
    else
    {
        animator.SetIKPositionWeight(AvatarIKGoal.LeftHand, 0);
        animator.SetIKRotationWeight(AvatarIKGoal.LeftHand, 0);
        animator.SetIKPositionWeight(AvatarIKGoal.RightHand, 0);
        animator.SetIKRotationWeight(AvatarIKGoal.RightHand, 0);
    }
}

```

}}

Отже, можна досить легко досягнути ефекту присутності у VR завдяки використанню інверсивної кінематики.

### 2.3. Hand-tracking.

Необхідно розуміти, що технологія відстежування рук не претендує повністю замінити контролери. Особливо в іграх або творчих застосунках, які вимагають високої точності, бо метод введення завдяки відстежуванню рук є неточним, в порівнянні з контролерами. Наразі розпізнаються тільки жести притискання, відтискання та притискання і утримання. Можливо інтегрувати руки для взаємодій, як клік, прокрутка, перетягування, повернення та вихід у системне меню (Рис. 4). Функція відстеження рук дозволяє переключатися на управління контролерами без необхідності виходу з додатку. Для взаємодії з UI елементами використовуючи руки, позиція руки керує курсором-вказівником, який веде себе як звичайний курсор контролера [9].



Pinch to Select.  
Pinch and Drag to Scroll



Look at your raised palm and pinch to return  
to System Menu

Oculus також опублікували рекомендації та найкращі практики по розробці управління руками в додатку [10]. Вони вказали, що існують ряд переваг використання Hand-Tracking технології:

- Управління руками є доступним, бо не потребує додаткового обладнання.
- На відміну від інших пристроїв введення, управління руками автоматично вмикається, як тільки одягнути гарнітуру.
- Більший ефект присутності та соціальної присутності, де є можливим використання реальних рук та жестів.
- Не потрібно нічого тримати в руках, і вони можуть вільно взаємодіяти з об'єктами.

Але також є деякі ускладнення. Всі очікують, що руки у віртуальній реальності будуть працювати так само як і в реальному житті, але це нереально з кількох причин. По-перше, через технічні обмеження камери завдяки яким відстежуються руки та те, що руки можуть перекриватися іншими об'єктами з реального світу. По-друге, об'єкти у віртуальній реальності, з якими взаємодіє користувач, не надають ніякої тактильної відповіді, до яких звикли при взаємодії з реальними об'єктами. По-третє, вибір жестів для певної дії без випадкового спрацювання, може бути дуже важким, оскільки руки формують різні пози навіть при звичайній розмові.

Для того, щоб увімкнути технологію відстеження рук, необхідно додати дозволи у AndroidManifest.

```
<uses-permission  
android:name="com.oculus.permission.HAND_TRACKING" />  
<uses-feature android:name="oculus.software.handtracking"  
android:required="false" />
```

Окрім цього, користувач обов'язково повинен увімкнути відстеження рук на своєму Oculus Quest в розділі експериментальних технологій. На сьогодні VRTK не додали підтримку Hand-Tracking технології, тож доведеться звертатися та працювати напряму з Oculus SDK. Перш за все потрібно додати до якорів лівої та правої руки префаб OVRHandPrefab. Тобто ієрархія сетапу для Oculus буде виглядати так:

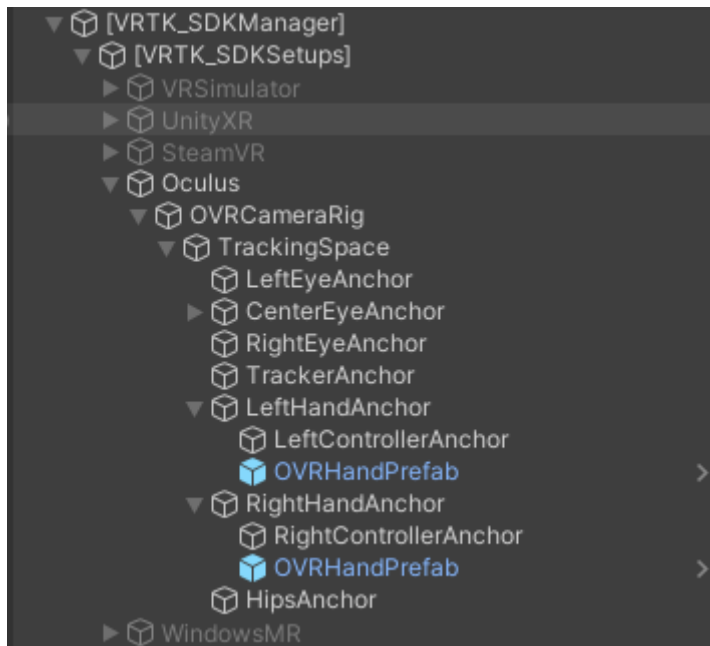


Рис. 5

Після цього вже можливо почати використовувати руки як метод введення.

Є можливість змінити зовнішній вигляд руки, але для цього обов'язково потрібні моделі рук з повним скелетом. Також можливо увімкнути колайдери, щоб фізично взаємодіяти з об'єктами оточення, або ж діставати відфільтровану позицію курсора вказівника. Прості додатки, що вимагають лише взаємодії клацання, можуть оброблювати позицію вказівника від рук,

як простого вказівного методу. При цьому жест натиску буде виконувати дії кліку.

Для того, щоб у будь-який момент додаток міг перевірити виявлення та відстеження рук, у компонента `OVRHand` є властивість `isTracked`, що дозволяє перевірити чи видно руки на даний момент і чи не відключено їх відстеження гарнітурою. Також надається властивість `HandConfidence`, яка вказує рівень впевненості у відстеження рук. Це має використовуватися у додатку, щоб контролювати відображення рук. Наприклад сховати моделі рук при низькому рівні впевненості, щоб позбутися некоректного відображення позиції рук. Приклад використання :

```
var hand = GetComponent<OVRHand>();
if(hand.HandConfidence == TrackingConfidence.High)
    gameObject.SetActive(true);
else
    gameObject.SetActive(false);
```

Щоб виявити чи виконується жест натиску вказівним пальцем та перевірити силу натиску, існують методи `GetFingerIsPinching()` та `GetFingerPinchStrength()` відповідно. Для цих методів необхідно вказувати константу пальця, яку необхідно перевірити. Існують такі константи: `Thumb`, `Middle`, `Ring`, `Pinky`, для кожної руки. Приклад використання :

```
var hand = GetComponent<OVRHand>();
bool isIndexFingerPinching =
hand.GetFingerIsPinching(HandFinger.Index);
float ringFingerPinchStrength =
hand.GetFingerPinchStrength(HandFinger.Ring);
```

А для перевірки впевненості у відстежуванні конкретного пальця, використовується `GetFingerConfidence()`, що також приймає константу пальця. Приклад використання:

```
var hand = GetComponent<OVRHand>();
TrackingConfidence confidence =
hand.GetFingerConfidence(HandFinger.Index);
```

Щодо вказівника, можна використовувати метод `PointerPose()` для кожної руки, та викликати `IsPointerPoseValid()` для перевірки чи правильна позиція вказівника. Приклад використання:

```
var hand = GetComponent<OVRHand>();
if(hand.IsPointerPoseValid) {

    Ray ray = new Ray(transform.position, hand.PointerPose.position);
    RaycastHit hit;
    //Створення променя від контролера до позиції вказівника
    if(Physics.Raycast(ray, out hit, maxDistance)){
        //Реалізація взаємодії з об'єктом на який вказує луч
    }
}
```

## 2.4. Оптимізація застосунку.

Оскільки VR використовує більше ресурсів ніж звичайні застосунки для рендеру на два дисплеї, оптимізація відіграє важливу роль у розробці. Одним з методів оптимізації є запікання світла.

Запікання світла (Light Baking) – процес обрахування інформації про освітлення для статичних сцен та світла[6]. Статичне світло можливо зробити

realtime, тобто таким самим як і динамічне, його вплив на все навколо буде обраховувати кожен кадр. Або ж попередньо запекти світло на саму сцену, що, очевидно, буде використовувати менше ресурсів. Запікання світла – найпоширеніший підхід для досягнення глобальної ілюмінації в іграх та 3д застосунках. При запіканні беруться текстури всіх об'єктів навколо світла, рушій обраховує вплив на ці текстури та зберігає цю інформацію у текстурі, що називається світлова карта (lightmap). Крім відносно низької обчислювальної вартості, запікання світла робить можливим досягти кращої якості освітлення, адже немає необхідності знижувати такі налаштування, як кількість відбивань світла та накладання певних фільтрів, бо після запікання вони не будуть впливати на головний процес.

Також, одним із способів оптимізації є використання рівня деталізації. Багато трикутників та вершин моделі у полі зору будуть непомітні для користувача. На великій відстані до точки огляду їх проекція на екран істотно менша ніж роздільна здатність пікселя дисплея. Тож немає необхідності навантажувати систему обрахуванням трикутників, що не помітні користувачеві[7]. Для вирішення цієї проблеми використовуються рівні деталізації (Level of detail). LOD передбачає зменшення кількості трикутників по мірі віддалення від точки огляду або за такими показниками, як важливість об'єкту, швидкість відносно точки огляду, тощо. Знижена візуальна якість моделі непомітна через невеликий вплив на зовнішній вигляд предмета при віддаленні або швидкому переміщенні. Рівні деталізації застосовуються не тільки для геометрії, а також для контролювання складності пікселя в шейдерах та застосовуються до текстур, але під назвою mipmaping, що використовує копії текстури але з різною роздільною здатністю.

## Висновки

Отже, оскільки сфера віртуальної реальності знаходиться у постійному розвитку, відповідно швидко з'являються нові методи для створення застосунків віртуальної реальності, що вирішують проблеми і помилки минулих методів або засобів. Також вони набагато розширюють та поглиблюють досвід віртуальної реальності, що залучає нових користувачів. Тому є сенс використовувати найактуальніші розробки.

У ході технічної сторони курсової роботи було розроблено проект для демонстрації можливості створення мультиплатформенного VR додатку. Також використано такі технології, як Hand-Tracking та Inverse Kinematics для більшого занурення користувача.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- 1) Сара ДіДжуліо. 3 Ways Virtual Reality Is Transforming  
[Електронний ресурс] / Сара ДіДжуліо // Інтернет портал “NBC News”. – 2017.  
[Електронний ресурс] URL: <https://www.nbcnews.com/mach/science/3-ways-virtual-reality-transforming-medical-care-ncna794871>
- 2) OpenVR Wiki  
[Електронний ресурс] URL: <https://github.com/ValveSoftware/openvr/wiki/API-Documentation>
- 3) VRTK  
Режим доступу до ресурсу: <https://vrtoolkit.readme.io/>
- 4) Wiki // Inverse Kinematics  
[Електронний ресурс] URL: [https://en.wikipedia.org/wiki/Inverse\\_kinematics](https://en.wikipedia.org/wiki/Inverse_kinematics)
- 5) Marko B. Popovic. Biomechatronics, 2019.  
[Електронний ресурс] URL: <https://www.sciencedirect.com/book/9780128129395/biomechatronics>
- 6) David Larsson, Autodesk Inc. Pre-computing Lightning in Games, 2010.  
[Електронний ресурс] URL: [https://cgg.mff.cuni.cz/~jaroslav/gicourse2010/giai2010-06-david\\_larsson-slides.pdf](https://cgg.mff.cuni.cz/~jaroslav/gicourse2010/giai2010-06-david_larsson-slides.pdf)
- 7) David Luebke, Level of Detail for 3D Graphics, 2003.  
[Електронний ресурс] URL: <http://index-of.co.uk/Game-Development/Programming/Level%20of%20Detail%20for%203D%20Graphics.pdf>
- 8) Oculus, Introducing Hand Tracking on Oculus Quest, 2019.  
[Електронний ресурс] URL: [https://www.oculus.com/blog/introducing-hand-tracking-on-oculus-quest-bringing-your-real-hands-into-vr/?locale=ru\\_RU](https://www.oculus.com/blog/introducing-hand-tracking-on-oculus-quest-bringing-your-real-hands-into-vr/?locale=ru_RU)
- 9) Oculus, Hand-Tracking Documentation for Unity, 2020.

[Электронный ресурс] URL:

<https://developer.oculus.com/documentation/unity/unity-handtracking/>

10) Oculus, Designing for Hands, 2020.

[Электронный ресурс] URL: <https://developer.oculus.com/design/hands-design-intro/>