

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

SINGAPORE

Virtual Reality for Immersive e-Learning

Submitted by: Low Jin Teng Jackson

Matriculation Number: U1922836D

Supervisor: Prof Seah Hock Soon

Examiner: Ast/P Mohamed M.Sabry

School of Computer Science & Engineering

A final year project report presented to the Nanyang Technological University in partial fulfilment of the requirements of the Degree of Bachelor of Engineering (Computer Science)

Table of Contents

Abstract	iv
Acknowledgements	v
List of Figures	vi
List of Tables	vii
Chapter 1 Introduction	1
1.1 Background	1
1.2 Project Specification	2
1.2.1 Hardware.....	2
1.2.2 Software.....	2
1.3 Scope and Objectives	3
1.3.1 Scope.....	3
1.3.2 Objectives	3
1.4 Organisation of the Report	4
Chapter 2 Literature Review	5
2.1 Overview of VR	5
2.2 VR Headsets	6
2.2.1 Mobile/Smartphone VR.....	6
2.2.2 Standalone/Wireless VR	7
2.2.3 Tethered/Desktop/PC VR	7
2.3 Immersive Learning	8
2.4 Hand Tracking.....	11
2.4.1 Camera Optical Tracking.....	11
2.4.2 LiDAR Laser Tracking	11
2.4.3 Hand Tracking in Oculus Quest 2	12
2.4.4 VR Haptic Gloves.....	13
2.5 Limitations of VR.....	14
2.5.1 High Costs	14
2.5.2 Impact to the Human Body	14
2.5.3 Addiction and Social Isolation.....	15
2.6 Rubik's Cube.....	15
Chapter 3 Implementation.....	17

3.1	Creating the Rubik's Cube	17
3.2	Adding Functionalities to the Cube.....	19
3.2.1	Basic Cube Notations	19
3.2.2	Tracking the Cube.....	20
3.2.3	Rotating the Entire Cube	23
3.2.4	Rotating the Layers	24
3.3	Starting Scene.....	26
3.4	Tutorial Scene	28
3.5	Image Cube	29
3.6	Hand Gestures	30
3.6.1	Setting Up	30
3.6.2	Recording Hand Gestures	31
3.6.3	Recognising Hand Gestures.....	32
3.6.4	Mapping the Gestures to the Functions	33
3.7	Adding Audio	34
	Chapter 4 Evaluation	35
4.1	Beta Testing.....	35
4.1.1	List of Participants	35
4.1.2	Questions	35
4.2	Results and Responses	36
4.2.1	Overall Opinion	36
4.2.2	Suggestions for Improvement.....	37
4.2.3	Interesting Observations	38
4.3	Limitations	38
4.4	Key Findings	39
4.4.1	Adequate Substitute but never the Real Thing	39
4.4.2	Balancing Costs and Rewards	39
4.4.3	Customisability and Scalability are Key.....	40
	Chapter 5 Conclusion.....	41
5.1	Conclusion.....	41
5.2	Recommendations for Future Work.....	41
	References.....	42
	Appendix.....	45

Abstract

This project explores the use of virtual reality (VR) in immersive e-learning by implementing a Rubik's Cube in VR to investigate its effectiveness as a learning tool. The cube was controlled entirely using hand gestures, and beta testing revealed that the virtual Rubik's Cube was not as effective as the physical cube due to accuracy and speed of recognition in detecting hand gestures. Nevertheless, it was found that there is a potential that users are able to develop muscle memory and there are areas where learning in VR can have an edge such as customization and scalability. The findings suggest that while there are limitations to using VR in certain contexts, it can still be a valuable tool for immersive e-learning with the potential for further development in specific areas.

Acknowledgements

I would like to express my sincere and heartfelt gratitude to my project supervisor, Professor Seah Hock Soon, for his invaluable guidance and support throughout the course of this project.

He has provided me with constant encouragement, constructive criticism, and his willingness to share his time and ideas have been instrumental in the development of this project. I am deeply grateful for his unwavering support and for always challenging me to think critically and pushing me to strive for a better project.

I would also like to extend my appreciation to my family and friends for their patience, encouragement and understanding during my challenging but fulfilling academic journey.

Thank you all for your contributions towards my academic and personal growth and this project would not have been possible without your support.

List of Figures

FIGURE 1: OCULUS QUEST 2 HEADSET	2
FIGURE 2: INFOGRAPHIC ON THE DIFFERENT USES OF VR.....	5
FIGURE 3: EXAMPLES OF MOBILE VR HEADSETS	6
FIGURE 4: EXAMPLES OF STANDALONE VR HEADSETS	7
FIGURE 5: EXAMPLES OF TETHERED VR HEADSETS	8
FIGURE 6: MEDICAL STUDENTS ENGAGING IN IMMERSIVE LEARNING	9
FIGURE 7: GAMEPLAY OF PIANOVISION	9
FIGURE 8: PHYSICALLY DISABLED PERSON ENJOYING THE VR EXPERIENCE	10
FIGURE 9: A VR SCHOOL ENVIRONMENT FROM PROJECT VOISS.....	10
FIGURE 10: GAMEPLAY OF UNPLUGGED.....	12
FIGURE 11: EXAMPLE OF A VR HAPTIC GLOVE.....	13
FIGURE 12: PICTURE OF A RUBIK'S CUBE.....	15
FIGURE 13: DIFFERENT VARIATIONS OF THE RUBIK'S CUBE.....	16
FIGURE 14: CREATING A SINGLE PIECE OF THE CUBE	17
FIGURE 15: THE ENTIRE 3X3 CUBE WITH RENAMED INDIVIDUAL PIECES.	17
FIGURE 16: BEFORE VS AFTER REMOVING THE INCORRECT SIDES	18
FIGURE 17: CREATING PLACEHOLDERS FOR THE RAYS	18
FIGURE 18: EXAMPLE OF SLICE MOVE AND DOUBLE LAYER TURN	20
FIGURE 19: CREATING LISTS TO TRACK EACH SIDE.....	20
FIGURE 20: CREATING LISTS OF FOR THE RAYS.....	21
FIGURE 21: GENERATING RAYS	21
FIGURE 22: ADDING THE FACES INTO TO THE LIST IF IT INTERSECTS WITH THE RAYS.....	22
FIGURE 23: RAYS IN DEBUG MODE.....	22
FIGURE 24: ROTATION GONE WRONG	23
FIGURE 25: EXAMPLE OF CUBE ROTATION	24
FIGURE 26: UPDATE() METHOD TO ROTATE THE CUBE.....	24
FIGURE 27: SELECTING THE ENTIRE LAYER	24
FIGURE 28: REVERTING THE PIECES BACK	25
FIGURE 29: FUNCTIONS TO ROTATE A LAYER OF THE CUBE	25
FIGURE 30: PASSING THE CORRECT PARAMETERS FOR EACH NOTATION.	26
FIGURE 31: FUTURISTIC-LOOKING ROOM.....	26
FIGURE 32: ADDING TABLE WITH DIFFERENT SIZED CUBES.....	27
FIGURE 33: FINAL STARTING SCENE	27
FIGURE 34: CUBES SCRAMBLED TO THE SAME CONFIGURATION	28
FIGURE 35: LEARNING TO SOLVE THE SECOND LAYER	28
FIGURE 36: IMAGE CUBE	29
FIGURE 37: SOLVING THE IMAGE CUBE.....	29
FIGURE 38: OCULUS INTEGRATION PACKAGE.....	30
FIGURE 39: HANDS VISIBLE IN UNITY GAME VIEW	30
FIGURE 40: CREATING GESTURE STRUCTURE.....	31
FIGURE 41: FUNCTION TO SAVE GESTURES	31
FIGURE 42: COPYING GESTURES CREATED.....	31
FIGURE 43: RECOGNISING GESTURES	32
FIGURE 44: ADDING CLICK SOUND WHEN LAYER ROTATES.....	34
FIGURE 45: EXAMPLE OF A SUPERCUBE.....	37

List of Tables

TABLE 1: BASIC CUBE NOTATIONS..... 19

TABLE 2: MAPPING OF GESTURES..... 34

TABLE 3: LIST OF PARTICIPANTS 35

TABLE 4: LIST OF QUESTIONS..... 36

Chapter 1 Introduction

1.1 Background

The idea of “Immersive experiences” may date back to the 1960s when the Sensorama, a multi-sensory machine that played 3D films along with stereo sound, aromas and wind was first invented [1].

Fast forward to today, 360° Videos, Virtual Reality (VR), Augmented Reality (AR) and Mixed Reality (MR), while different in their definitions, are all classified as Extended Reality (XR) which aims to provide immersive experiences to the user by creating an emotional response through the five senses.

With the COVID-19 pandemic in 2019, the rise of XR technology was significantly accelerated as businesses and schools turned to remote work and consumers become more accepting and willing to adopt XR in everyday usage. The global XR market was valued at 38.3 billion USD in 2022 and is expected to reach 394.8 billion in 2030 [2].

XR is used in many industries including but not limited to gaming, healthcare, education, marketing, military, real estate, and even for charity. According to a 2019 Survey by Perkins Coie, Education is one of the biggest industries where XR is most applicable and where most investment are directed [3].

VR in Education, or more commonly known as Immersive Learning is an approach based on decades of neuroscience research, which indicated that the brain treats VR experiences just like it would treat real life. It combines the sense of presence of VR with advanced learning theory, data science and spatial design to improve effectiveness and user engagement. There are many successful examples of VR in Education including simulators for flight and medical training, virtual campuses, virtual museums and field trips.

In spite of this, there are still limitations to Immersive Learning when compared to Traditional Learning such as the high cost, accessibility, flexibility and potential health issues. As such, it is imperative that we understand the advantages and disadvantages in using XR in learning and extend this knowledge to improve the development of XR applications to provide a more fulfilling user experience.

1.2 Project Specification

1.2.1 Hardware

For this project, I will be using my own Oculus/Meta Quest 2 for the implementation and testing of the VR game. The Oculus Quest 2 is a standalone VR headset developed by Oculus, a subsidiary of Facebook.

It features a resolution of 1832 x 1920 pixels per eye, which provides a high level of clarity and detail in VR environments. With six degrees of freedom (6DoF) tracking, it allows for precise and responsive tracking of the user's head and hand movements in VR.

One notable feature of the Oculus Quest 2 is its hand tracking capability, which enables natural and intuitive interaction with virtual environments using only the user's hands. This feature uses advanced computer vision algorithms to track the user's hand movements and gestures, allowing for a wide range of interactions such as picking up and manipulating objects.



Figure 1: Oculus Quest 2 Headset

1.2.2 Software

The software I will be using for this project is mainly Unity version 2021.3.16f1. Unity is a cross-platform game engine and development platform that is widely used for interactive and immersive experiences, including VR applications.

The Oculus Integration Package will also be utilised to enable the hand tracking feature specific to the Oculus platform.

1.3 Scope and Objectives

1.3.1 Scope

The scope of the project is to investigate the pros and cons of using Virtual Reality for e-Learning and provide possible solutions and innovations that will be useful in future VR application/game development.

In particular, I will be designing a Rubik's Cube tutorial that can be controlled entirely through hand gestures on the Oculus Quest 2 headset. The Rubik's Cube is a well-known and widely popular puzzle game, which requires some level of muscle memory to help memorise certain algorithms and moves. By using hand gestures to control the cube, we can simulate some aspects of the actual physical moves and explore the effectiveness of using hand gestures to help users with memory retention.

The project will involve designing and developing the Rubik's Cube game environment in VR, creating and integrating the hand gesture recognition system for controlling the game, and testing the game to ensure its functionality and usability.

1.3.2 Objectives

The objectives of the project are defined as follows:

- a. To develop a functional Rubik's Cube tutorial in Virtual Reality and evaluate its effectiveness.
- b. To compare the key differences of learning via Virtual Reality compared to Traditional Learning in the real world.
- c. To investigate the effectiveness of using hand gestures in Virtual Reality and its ability to help users develop muscle memory.
- d. To provide possible improvements and innovations to the learning experience using Virtual Reality.

1.4 Organisation of the Report

The report is organised into five chapters.

Chapter 1 provides an introduction to the project, outlining the scope and objectives, and presents an overview of the background and the rest of the report.

Chapter 2 presents a review of the existing literature on VR, immersive e-learning, hand gesture recognition, and the Rubik's Cube. It discusses the relevant theories and concepts, as well as the current state-of-the-art research in these fields.

Chapter 3 provides a detailed description of the implementation of the Rubik's Cube Tutorial application using Unity and the Oculus Quest 2 headset. It covers the design of the application, the development process, and the key features and components of the application.

Chapter 4 presents the testing and evaluation of the application, including the results and findings, user feedback, limitations as well as areas for improvement.

Chapter 5 concludes the report by offering a reflection on the project and provides recommendations for future work.

Chapter 2 Literature Review

2.1 Overview of VR

Virtual Reality (VR) is a rapidly evolving field that has gained significant attention in recent years, due to increasing consumer and enterprise demand, growing use cases, and a booming ecosystem [4]. From Figure 2, we can see that VR has been adopted in many sectors today, including medicine, architecture, education and even in the military [5].

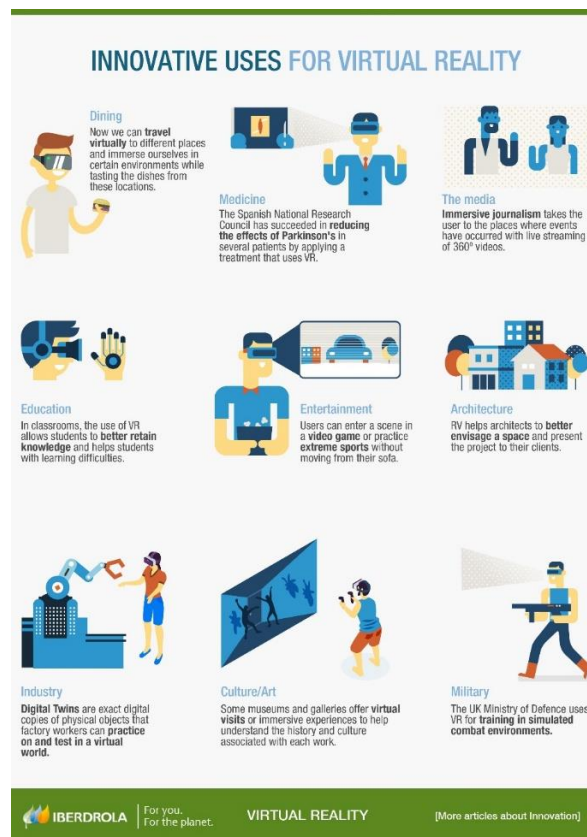


Figure 2: Infographic on the different uses of VR

VR works because of a physiological concept known as stereopsis, which is our ability to perceive depth from the horizontal differences in the image each eye receives when we look at something [6]. By using computer-generated 3D images and having it change instantly as the point of view of the user is changed [7], the VR headset is able to deliver a sense of presence and create visual immersion for the user. While human vision provides the most information to the brain, the synchronisation of other senses

especially hearing and touch through spatial audio and haptic feedback respectively further contributes to the immersive experience.

2.2 VR Headsets

There is a variety of VR headsets available providing a range of different VR experiences, but they can be broadly categorised into the following 3 main types [8]:

2.2.1 Mobile/Smartphone VR

Mobile/Smartphone VR like its name suggests, uses smartphones to provide a VR experience. Users simply have to slide their smartphones into the headset with built-in lenses which creates the sense of depth required for the VR experience.

The main benefit of mobile VR headsets is the relatively inexpensive price tag which can be attractive to users who are looking for a basic introduction to VR. Most mobile VR headsets, however, only have rotational tracking with three degrees of freedom (3DoF) and this applies to the controllers as well [9].

As such, they are often used for seated content such as watching videos or casual gaming and provides a limited immersion experience compared to the other types of VR headsets.

From Figure 3, some examples of mobile VR headsets include (from left to right) the Samsung Gear VR, Google Daydream View 2, Xiaomi VR Play 2, and the Google Cardboard.



Figure 3: Examples of Mobile VR headsets

2.2.2 Standalone/Wireless VR

Standalone/Wireless VR headsets have built-in processors, sensors, batteries, storage memory and displays, and are sometimes referred to as all-in-one VR headsets. Apart from charging the battery, they do not require any external physical connection, which allows the user to use it anywhere and makes it easier for them to find a large enough space.

Generally, standalone VR headsets are much less powerful than tethered VR headset, offering lower quality graphics and lower refresh rates, but they make up for it by being more affordable and dynamic.

From Figure 4, some examples of standalone VR headsets include (from left to right) the Lenovo Mirage Solo, HTC VIVE Focus, Pico Neo, and the Oculus GO.



Figure 4: Examples of standalone VR headsets

2.2.3 Tethered/Desktop/PC VR

Tethered/Desktop/PC VR refers to the headsets that are required to be physically connected to a computer. They generally provide the most immersive and highest quality experiences as they can utilise the capabilities of a powerful high-end PC.

The biggest drawback of such headsets is the restricted movement and the space required due to the constant cable connection required. While some tethered VR headsets may be similarly priced with standalone VR headsets, the additional external components required to make it work often makes it the premium option.

From Figure 5, some examples of tethered VR headsets include (from left to right) the Oculus Rift, HTC VIVE Pro, Lenovo Explorer and Samsung Odyssey.



Figure 5: Examples of Tethered VR headsets

2.3 Immersive Learning

Immersive Learning is a form of education that involves creating an interactive and engaging environment for learners, through the use of virtual and augmented reality to create realistic simulations of real-world environments. These simulations can be used to provide learners with hands-on experience in a safe and controlled environment, allowing them to practice and improve their skills without any risk.

For example, medical students from the National University of Singapore (NUS) Yong Loo Lin School of Medicine, have begun using VR to experience the process of patient safety and immersion in operating theatre procedures. The system, also known as the Patient Safety as Inter-Professional Training (PASS-IT), is a digital gamified environment that allows students to learn about hands-on technique in the operating theatre which is otherwise a highly restricted environment [9].



Figure 6: Medical students engaging in immersive learning

In addition to the healthcare sector, immersive learning can also be applied to various other fields, such as aviation, engineering, architecture, and even music. PianoVision is an augmented reality (AR) game which uses Passthrough AR and hand tracking to accelerate piano learning, and even allow users to upload their own sheet music. In 2020, the Royal College of Music in London also began developing VR technology for students to practice performing in iconic concert venues [10].

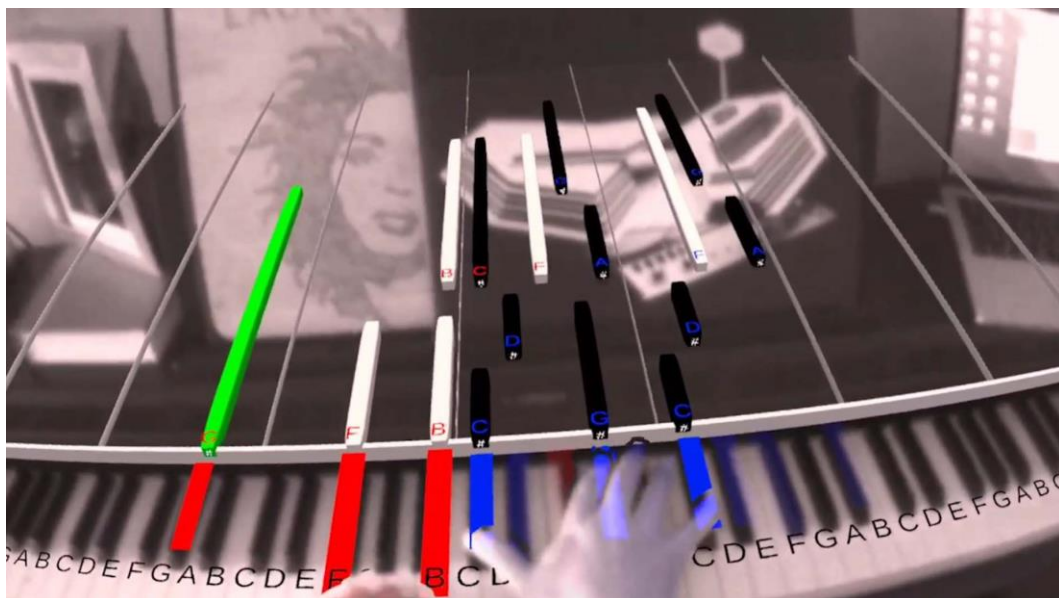


Figure 7: Gameplay of PianoVision

Immersive learning can also help individuals with disabilities or limited physical mobility to have access to experiences that may not be possible in the real world, such as snorkelling and rock climbing [11].



Figure 8: Physically disabled person enjoying the VR experience

For individuals with learning disabilities or autism, immersive learning can help to nurture their social skills and improve social competencies. In 2018, the United States Department of Education invested \$2.5 million USD to launch a project called Virtual Reality Opportunities to Implement Social Skills (VOISS), where students are presented with social stories and situations across multiple school-based environments [12].



Figure 9: A VR school environment from project VOISS

No matter the topic or learning outcome, the beauty of immersive learning is that learning designers are able to create suitable virtual environments for learning to take place without the barriers that may exist if they tried to recreate those environments in the real world [13].

2.4 Hand Tracking

Hand tracking technology enables users to interact with virtual environments with their hands and without the need for physical controllers. It has the potential to revolutionise the way we interact with VR and AR, making it more intuitive, immersive and accessible.

Hand tracking works by using the headset cameras, LiDAR array, or external sensor stations to track the hand's position, depth, pace and orientation, before analysing the data and translating it into a virtual, real-time representation [14].

There are two main types of hand tracking in virtual reality namely camera optical tracking and LiDAR laser tracking:

2.4.1 Camera Optical Tracking

Camera optical tracking is when the tracking originates from a camera array and it is bounded by the field of view (FOV) of the cameras, the sensitivity to light, and the cameras' capabilities to capture motion (frame rate). Optical tracking uses the difference in pixels to determine hand movements and generally requires more processing time to translate into virtual hands.

2.4.2 LiDAR Laser Tracking

Light Detection and Arranging, also known as LiDAR, pulses lasers to determine the distance, position and movement of the environment relative to the hands. Compared to optical tracking, this method is more precise as lasers are faster than a camera's assessment of pixels, and the bounce-back of the laser into the LiDAR sensor is able to immediately generate a 3D representation of what is tracked. Unlike optical tracking, the quality is also not lost based on the distance between the camera and the hands.

2.4.3 Hand Tracking in Oculus Quest 2

In 2019, Meta first launched the hand tracking feature on the Oculus Quest/Quest 2 and upgraded to hand tracking 2.0 in April 2022. The upgraded system can detect gestures such as high fives and clapping and improved how the system sensed movement if part of the hands were blocked from the headset's onboard camera.

The Oculus Quest uses “inside-out” tracking where tracking originates from the headset and four cameras placed on strategic spots to determine the user's position. Hand tracking is integrated into the operating system and many first party Quest applications, allowing users to access system menus and navigate around with ease without the use of controllers [15].

As a relatively new feature, there are not many third-party applications or games at the moment that completely utilises hand tracking, but the future of hand tracking in VR is promising as it is able to create experiences that are not possible with the support of controllers. For example, the game Unplugged is a rhythm game that uses precise hand/finger tracking to allow users to play songs on a virtual air guitar [16].

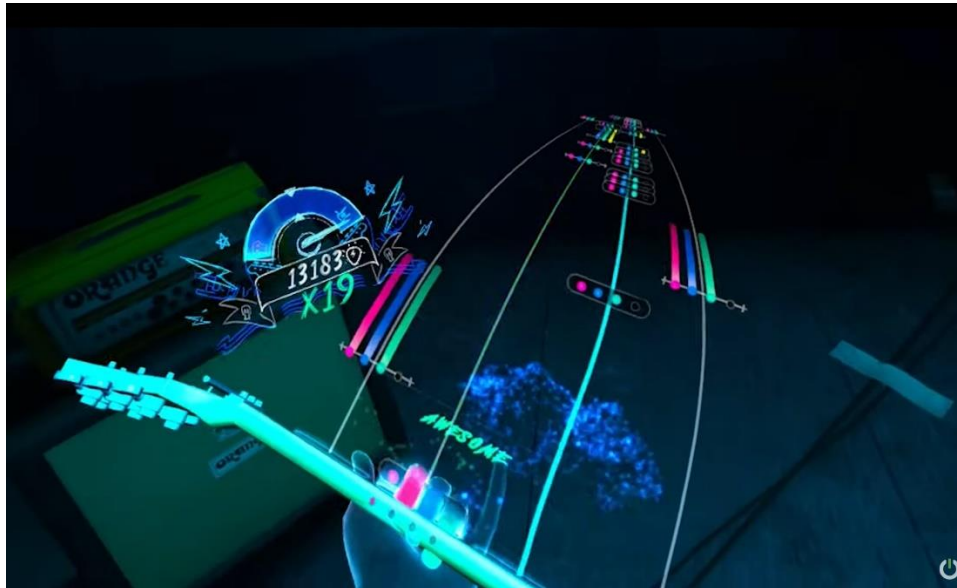


Figure 10: Gameplay of Unplugged

2.4.4 VR Haptic Gloves

An alternative to the usual camera optical and LiDAR tracking is the wearable technology known as VR haptic gloves. In addition to the regular hand tracking feedback, VR haptic gloves offer more advanced feedbacks described below [17]:

- a) **Vibrotactile Feedback:** Provides range of sensations, from light touches to rough textures, allowing users to feel cues, impacts and vibrations.
- b) **Force Feedback:** Emulates the feeling of object density and size by applying resistance through different magnetic friction brakes built into the glove.
- c) **Contact Feedback:** Recreates the sensation of touching objects in the VR space by using shape memory alloy coils to apply force to the ball of the fingers.
- d) **Temperature Feedback:** Allows the user to feel temperature differences through the use of thermoelectric devices (TEDs) embedded in the gloves. The user's body temperature is also measured and factored in, and the sensors are capable of simulating a temperature difference of up to 10 degrees in less than half a second [18].

While VR haptic gloves are able to provide a much more immersive experience, they come at a very costly price and are usually not affordable to a regular consumer. These gloves are more often used at a corporate level where they can be highly beneficial in simulated trainings in high-risk scenarios.



Figure 11: Example of a VR Haptic Glove

2.5 Limitations of VR

Having understood the basics of VR and the benefits it is able to provide, we will now analyse some of the limitations in using VR for Immersive e-Learning:

2.5.1 High Costs

The cost of using VR for Immersive e-Learning can be extremely high. Firstly, to enjoy the experience of VR, the user has to own a headset and a mid-range standalone VR headset can cost around \$300 to \$600. Secondly, the development of a custom VR training program has an average cost of \$50,000 to \$150,000 USD or more per project [19]. Even the simplest VR application and games can easily cost more than \$5,000 USD [20]. Thirdly, there will be miscellaneous and post-development costs involved such as bug fixes, maintenance of servers, updates which further adds to the cost of implementing a VR training programme.

2.5.2 Impact to the Human Body

The greatest concern with using VR for immersive learning is perhaps the negatives impacts to the human body. “Virtual Reality Sickness” refers to the symptoms associated with exposure to VR, and these include nausea, dizziness, sweating, loss of balance etc. Following a VR session, a temporary change can also be induced in the person’s sensory, motor and perceptual abilities, and affect their manual dexterity or ability to orientate their body [21].

The cause of these symptoms is due to the way VR affects the eye-brain connection as we are projecting something onto the eyes that looks far away, when it is in fact only a few centimetres away [22]. This is also known as the “vergence-accommodation conflict” and the seriousness and long-term effects of it has not yet been scientifically determined.

Other health concerns include myopia or short-sightedness and impact to hearing from long exposure to extremely close screens and loud volume.

2.5.3 Addiction and Social Isolation

Another limitation is using VR for immersive learning is the risk of addiction and social isolation. An immersive experience is a double-edged sword as it makes it easy for users to become absorbed and spend excessive time in the virtual environment, neglecting their real-life responsibilities and social interactions. Over time, this can lead to feelings of loneliness, social isolation, and a disconnection to reality.

While VR can be used to satisfy social needs and has been used to help individuals with underlying conditions, such as untreated social anxiety, the overuse of it may negatively affect the way we fulfil our needs and makes us withdraw physically from society [23].

2.6 Rubik's Cube

As my project involves developing a Rubik's Cube tutorial in the VR environment, this section discusses the origins and basic information about the Rubik's Cube:

The Rubik's Cube is a 3D combination puzzle invented in 1974 by Hungarian sculptor and professor of architecture Ernő Rubik. Originally known as the Magic Cube, the puzzle consists of 26 smaller cubes that can be rotated along each of three axes. The goal is to return the cube to its original state with each of the six faces displaying a single colour.



Figure 12: Picture of a Rubik's Cube

Research on the Rubik's Cube has focused on its mathematical properties, its algorithms, and its use in education and cognitive training. For example, mathematicians have studied the number of possible combinations of the Rubik's Cube which is over 43 quintillion.

In terms of algorithms, numerous methods have been developed for solving the Rubik's Cube, ranging from simple beginner's methods to advanced techniques used in speed cubing competitions. These algorithms typically involve a series of steps that aim to move specific pieces into desired positions while preserving the orientation of the other pieces.

The Rubik's Cube has also been used in education and cognitive training, as it can help improve memory, reflexes, problem solving and concentration [24]. Additionally, the Rubik's Cube has been used as a tool for teaching geometry and group theory in classrooms [25].

The Rubik's Cube has spawned many variations and extensions since its invention in the 1970s. One of the earliest extensions was the Rubik's Revenge, which is a 4x4 version of the original cube. Other variations include the Professor's Cube, which is a 5x5 cube, the V-Cube 6 and 7, which are 6x6 and 7x7 respectively, and the Megaminx, which is a dodecahedron-shaped puzzle with 12 sides. Additionally, there are many other shape-modified puzzles that are based on the mechanics of the Rubik's Cube, such as the Pyraminx, the Skewb, and the Square-1. These variations and extensions of the Rubik's Cube offer new challenges and opportunities for puzzle enthusiasts to explore and solve.



Figure 13: Different variations of the Rubik's Cube

Overall, the Rubik's Cube is a popular and enduring puzzle that has inspired a wealth of research and interest in both its mathematical properties and its applications in education and cognitive training.

Chapter 3 Implementation

In this chapter, I will describe the detailed implementation process and challenges faced in designing a VR Rubik's Cube tutorial using Unity.

3.1 Creating the Rubik's Cube

As there were no free Rubik's Cube asset available in the Unity Asset Store at the time of writing, we had to create it from scratch. The first step was to create a 1x1x1 cube and attach a smaller surface to each side of the cube. Materials of different colours were created and applied to each surface to create a single piece of the cube.

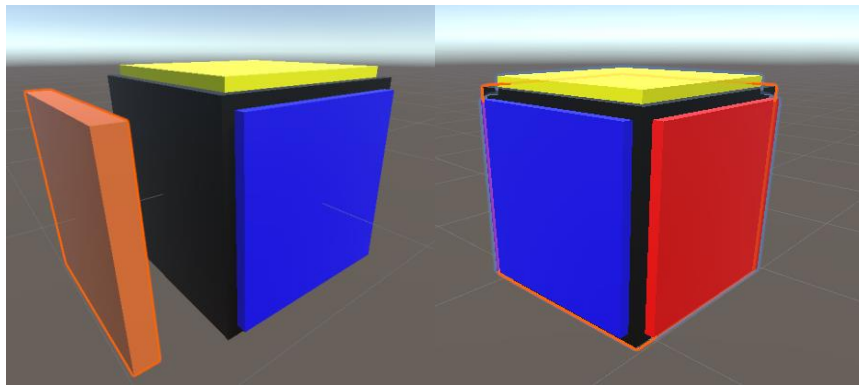


Figure 14: Creating a single piece of the cube

The piece was then duplicated 26 times to create the 3x3 cube and the middle piece was removed as it will never be visible no matter how the cube was rotated. Each individual piece was then renamed according to its position in the cube so that it can be easily identified and used later.

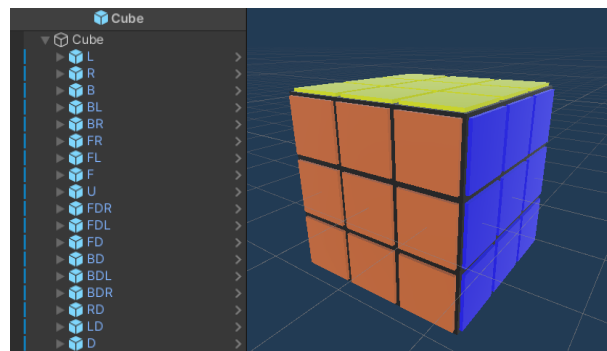


Figure 15: The entire 3x3 Cube with renamed individual pieces.

As the same piece was duplicated to create the entire cube, we have to ensure that only the correct sides of the pieces are visible when the cube is rotated. This can be done by making the small surface of the incorrect sides (sides facing inwards) inactive, which disables the components of the GameObject, including attached renderers, colliders, rigidbodies, and scripts. Each of these surfaces were also put in a new layer called “Faces” which we will use later for ray casting to track the faces of the cube.

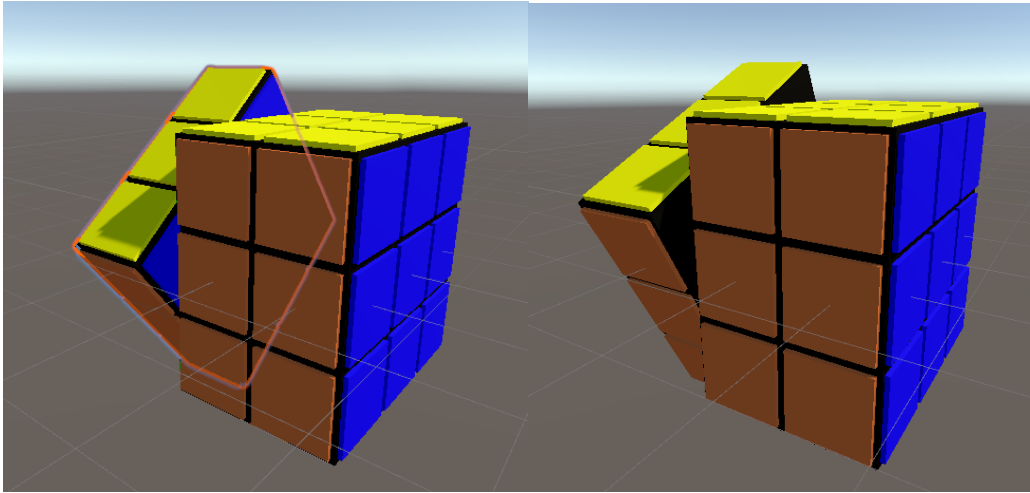


Figure 16: Before vs After removing the incorrect sides

In order to track the position of each piece of the cube as it rotates, we will cast 9 rays on each side of the cube. To do so, we first create six empty GameObject “Rays” and position each of them slightly in front of the middle piece on each side of the cube. The rest will be done via a C# script which will be explained later in the report.

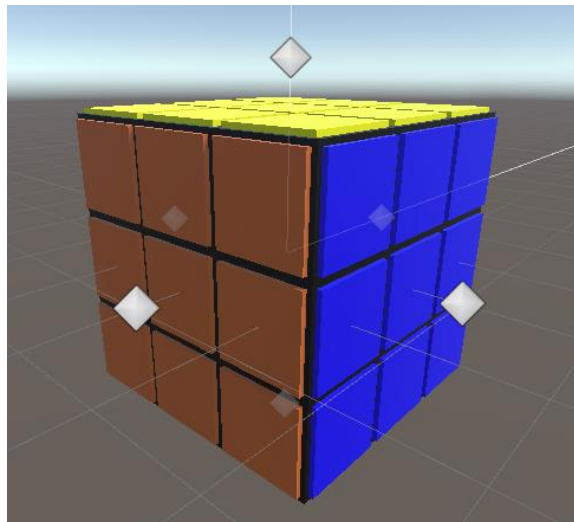


Figure 17: Creating placeholders for the rays

3.2 Adding Functionalities to the Cube

3.2.1 Basic Cube Notations

Before deciding which sides of the cube we want to track, it is important to first understand the basic notations that describe the rotation of the cube in the table below [26]:




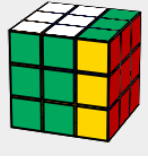
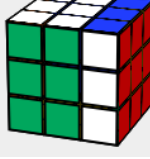

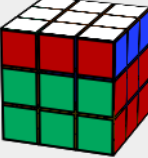



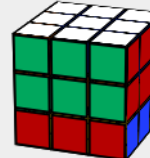
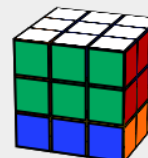





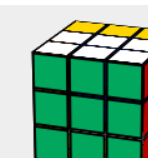
Notation	Cube after performing rotation		
Front: F, F', F2			
Right: R, R', R2			
Up: U, U', U2			
Down: D, D', D2			
Left: L, L', L2			
Back: B, B', B2			

Table 1: Basic cube notations

From Table 1, we can see that a letter by itself refers to a clockwise (relative to the face) rotation by 90 degrees, and a letter followed by an apostrophe is a counterclockwise rotation by 90 degrees. A letter followed by a “2” simply means a double turn or a rotation by 180 degrees.

The most basic method to solve a Rubik’s cube also known as the “Beginner’s Method” only requires the knowledge of these 6 letters (F, R, U, D, L, B) and their variations. Slice moves which involve turning the middle layer of the cube and double layer turns are not required and will not be implemented in this tutorial.

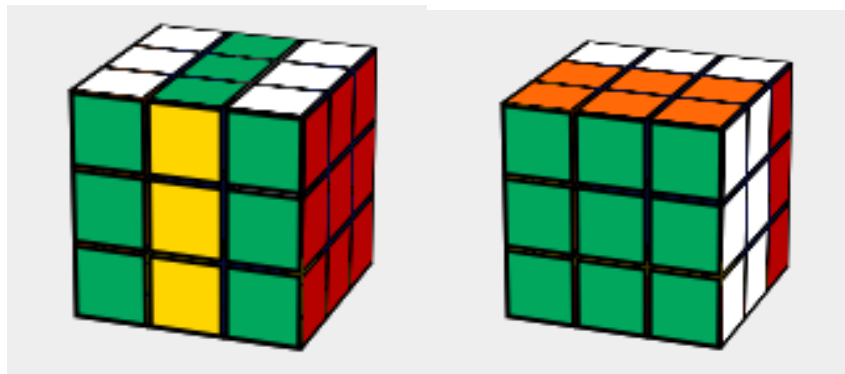


Figure 18: Example of slice move and double layer turn

3.2.2 Tracking the Cube

Now that we have decided to implement only 6 types of rotations, we only have to track 6 sides of the cube instead of each individual 26 positions. To do so, we first attached a C# Script “CubeState” to the cube and created 6 list of GameObjects.

```
public class CubeState : MonoBehaviour
{
    public List<GameObject> front = new List<GameObject>();
    public List<GameObject> back = new List<GameObject>();
    public List<GameObject> up = new List<GameObject>();
    public List<GameObject> down = new List<GameObject>();
    public List<GameObject> left = new List<GameObject>();
    public List<GameObject> right = new List<GameObject>();
}
```

Figure 19: Creating lists to track each side

The list is not yet populated, but will ideally contain 9 GameObjects or pieces each, and each piece can belong to up to 3 sides at once (for the corners). The next step is to cast rays from the placeholders we created previously. We first

referenced the transforms of the placeholders and created a list of GameObjects to store the rays for each of the sides.

```
public class CubeTrack : MonoBehaviour
{
    public Transform rUp;
    public Transform rDown;
    public Transform rLeft;
    public Transform rRight;
    public Transform rFront;
    public Transform rBack;

    private List<GameObject> frontRays = new List<GameObject>();
    private List<GameObject> backRays = new List<GameObject>();
    private List<GameObject> upRays = new List<GameObject>();
    private List<GameObject> downRays = new List<GameObject>();
    private List<GameObject> leftRays = new List<GameObject>();
    private List<GameObject> rightRays = new List<GameObject>();
}
```

Figure 20: Creating lists of for the rays

Next, we manually coded the direction for each of the ray's placeholder or starting position to ensure that they are pointing towards the cube and passed the transform and the direction as parameters into another method to generate the rays.

```
void SetRayTransforms()
{
    upRays = GenerateRays(rUp, new Vector3(90, 90, 0));
    downRays = GenerateRays(rDown, new Vector3(270, 90, 0));
    leftRays = GenerateRays(rLeft, new Vector3(0, 180, 0));
    rightRays = GenerateRays(rRight, new Vector3(0, 0, 0));
    frontRays = GenerateRays(rFront, new Vector3(0, 90, 0));
    backRays = GenerateRays(rBack, new Vector3(0, 270, 0));
}

6 references
List<GameObject> GenerateRays(Transform rayTransform, Vector3 direction)
{
    int count = 0;
    List<GameObject> rays = new List<GameObject>();

    for (int y = 1; y > -2; y--)
    {
        for (int x = -1; x < 2; x++)
        {
            Vector3 startPos = new Vector3(rayTransform.localPosition.x + x,
                rayTransform.localPosition.y + y,
                rayTransform.localPosition.z);

            GameObject rayStart = Instantiate(empty, startPos, Quaternion.identity, rayTransform);
            rayStart.name = count.ToString();
            rays.Add(rayStart);
            count++;
        }
    }

    rayTransform.localRotation = Quaternion.Euler(direction);
    return rays;
}
```

Figure 21: Generating rays

Using a double for loop, we generated 9 rays for each of the sides and passed the piece that each ray hits back into the CubeState lists we created earlier. A layerMask was used so that the ray will ignore all other colliders other than the faces of the cube. With this, we are able to call this method after every rotation to track the new position of all the pieces.

```
public List<GameObject> TrackFace(List<GameObject> rayStarts, Transform rayTransform)
{
    List<GameObject> facesHit = new List<GameObject>();

    foreach (GameObject rayStart in rayStarts)
    {
        Vector3 ray = rayStart.transform.position;
        RaycastHit hit;

        if (Physics.Raycast(ray, rayTransform.forward, out hit, Mathf.Infinity, layerMask))
        {
            Debug.DrawRay(ray, rayTransform.forward * hit.distance, Color.yellow);
            facesHit.Add(hit.collider.gameObject);
        }
        else
        {
            Debug.DrawRay(ray, rayTransform.forward * 1000, Color.green);
        }
    }

    return facesHit;
}
```

Figure 22: Adding the faces into to the list if it intersects with the rays

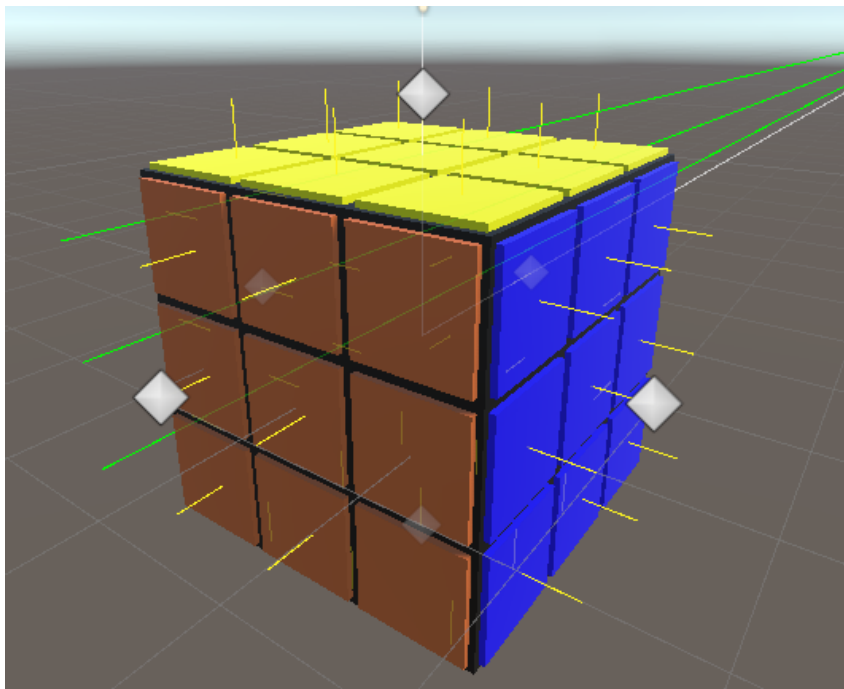


Figure 23: Rays in debug mode

3.2.3 Rotating the Entire Cube

Initially, the plan was to allow the user to grab the cube and rotate it with their hands as they would do in real life, to create the immersive experience. While this could be easily implemented, the problem arises when rotating each layer of the cube as it becomes harder to track and calculate the angle to rotate each piece in, making the problem a lot more complex.

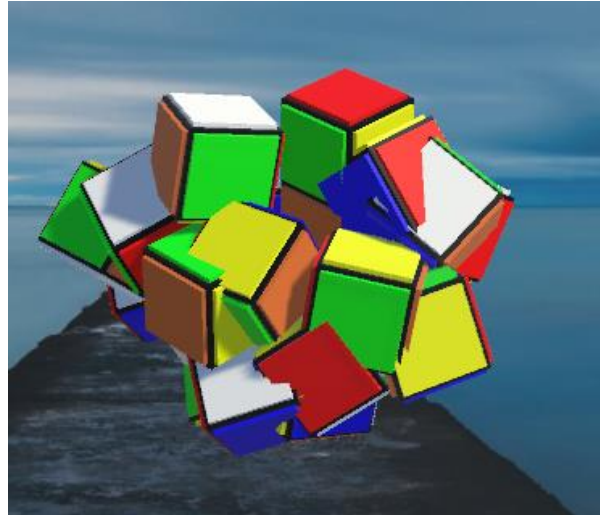


Figure 24: Rotation gone wrong

As such, I have decided to keep this as a possible future improvement instead and use hand gestures to rotate the cube in this implementation. Rotating the cube is simple to implement as I restricted the rotation to 90 degrees in only four possible directions (up, down, left, right).

To do so, we first required 2 flags, one to check if the cube was rotating and the other to check if any of the layers were rotating. This is because if the cube was to rotate again when it is already rotating, it may result in the image shown in Figure 24. Once we have ensured that the cube is not rotating, we can change the transform of a dummy GameObject which has the same transform as the original cube.

```

public void LeftSwipe()
{
    if (!CubeState.rotating)
    {
        if (!swiping)
        {
            swiping = true;
            target.transform.Rotate(0, 90, 0, Space.World);
        }
    }
}

```

Figure 25: Example of cube rotation

In every frame, the Update() method will be called to check if the cube's transform is equal to the dummy's transform and will rotate the former to the latter at a specified speed if they are not equal.

```

void Update()
{
    if (transform.rotation != target.transform.rotation)
    {
        var step = speed * Time.deltaTime;
        transform.rotation = Quaternion.RotateTowards(transform.rotation, target.transform.rotation, step);
    }
}

```

Figure 26: Update() method to rotate the cube

3.2.4 Rotating the Layers

Rotating the different layers of the cube was perhaps the most complex part of the project which took up the most amount of time. After many trials and errors, I decided to settle on a simple rotation mechanic, allowing users to only perform moves outlined in Chapter 3.2.1.

The first step is to create the ability to select the entire layer. We can do so by making the 8 pieces the child of the centre piece such that the entire layer can rotate together using the centre piece as the pivot.

```

public void SelectLayer(List<GameObject> cubeSide)
{
    foreach (GameObject face in cubeSide)
    {
        if (face != cubeSide[4])
        {
            face.transform.parent.transform.parent = cubeSide[4].transform.parent;
        }
    }
}

```

Figure 27: Selecting the entire layer

After rotating the layer, we must ensure that the pieces are returned back to their original hierarchy.

```
public void DeselectLayer(List<GameObject> cubeSide, Transform original)
{
    foreach (GameObject face in cubeSide)
    {
        if (face != cubeSide[4])
        {
            face.transform.parent.transform.parent = original;
        }
    }
}
```

Figure 28: Reverting the pieces back

Rotation of the layer is performed similarly to how we rotate the entire cube. We select the entire layer, calculate the target rotation, and set the “rotationStart” flag to true. In every frame, the Update() method will check the flag and perform the rotation when the flag is true. Once the rotation is performed, we will deselect the layer, track the cube again to update the new positions, and reset all the flags to false.

```
public void StartRotation(List<GameObject> side, float angle)
{
    cubeState.SelectLayer(side);
    Vector3 localForward = Vector3.zero - side[4].transform.parent.transform.localPosition;
    targetQuaternion = Quaternion.AngleAxis(angle, localForward) * transform.localRotation;
    activeSide = side;
    rotationStarted = true;
}

1 reference
private void RotateLayer()
{
    var step = speed * Time.deltaTime;
    transform.localRotation = Quaternion.RotateTowards(transform.localRotation, targetQuaternion, step);

    if (Quaternion.Angle(transform.localRotation, targetQuaternion) <= 1)
    {
        transform.localRotation = targetQuaternion;
        cubeState.DeselectLayer(activeSide, transform.parent);
        cubeTrack.TrackCube();
        CubeState.rotating = false;
        rotationStarted = false;
    }
}
```

Figure 29: Functions to rotate a layer of the cube

Finally, we simply have to pass in the parameters of the side and the angle to rotate by, for each of the 18 basic notations. With that, we have created a functional Rubik's Cube in Unity.

```
public void MoveByNotation (string move)
{
    if (!CubeState.rotating)
    {
        cubeTrack.TrackCube();
        CubeState.rotating = true;
        if (move == "U")
        {
            RotateLayer(cubeState.up, -90);
        }
        if (move == "U'")
        {
            RotateLayer(cubeState.up, 90);
        }
        if (move == "U2")
        {
            RotateLayer(cubeState.up, -180);
        }
    }
}
```

Figure 30: Passing the correct parameters for each notation.

3.3 Starting Scene

With a functional Rubik's cube, we can finally begin setting up our scene in Unity. The starting scene was designed using a free online asset to create a futuristic-looking room [27]. The benefit of using an asset instead of creating our own room is that we do not have to worry about the scale of the objects which is very important in VR.

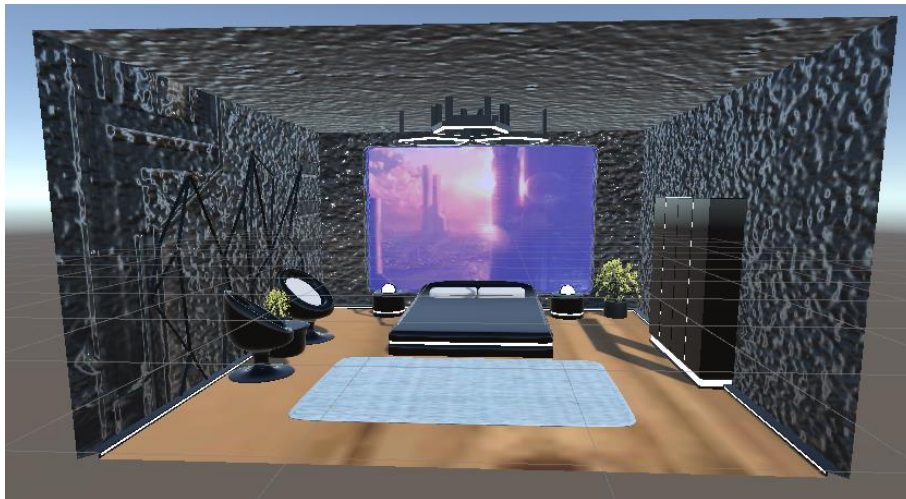


Figure 31: Futuristic-looking room

A table was added into the scene and cubes of different sizes were placed on it. The different sized cubes, 2x2, 4x4 and 5x5, were made similarly to the 3x3 cube but the

functionalities were not added as there would be more layers to track and manipulate which requires a significant amount of time. As such, only the original Rubik's cube and the 'Image Cube' which we will discuss further in Chapter 3.5 are functional in this implementation.



Figure 32: Adding table with different sized cubes.

Finally, we added an interface to guide the user on what to do, readjusted the camera starting position and created a level manager to switch between different scenes. User can now choose between the original Rubik's cube for the tutorial or the 'Image Cube' for practice.

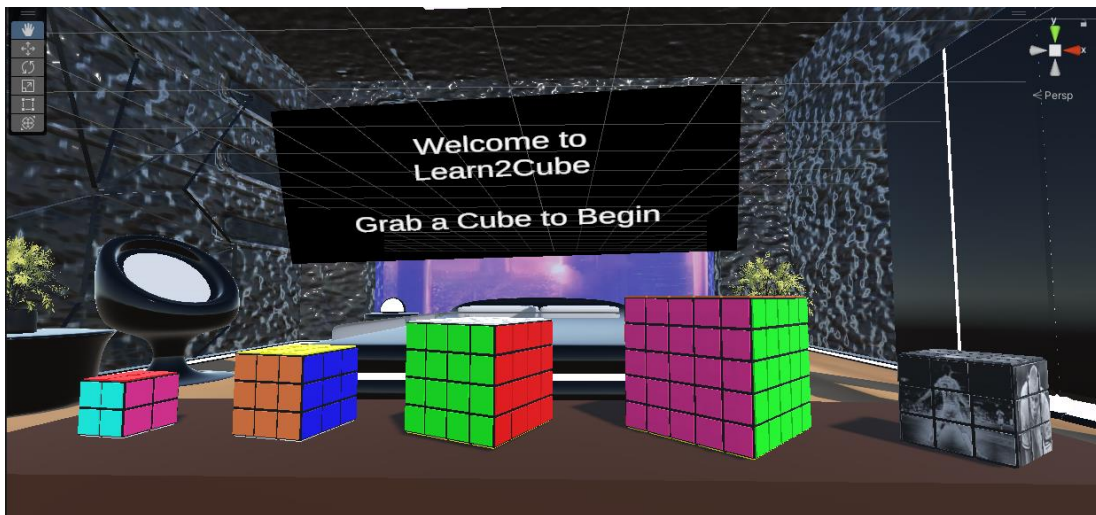


Figure 33: Final starting scene

3.4 Tutorial Scene

The tutorial for the Rubik's Cube is taught by using a much larger Rubik's Cube in the background with instructions displayed on what to do. Users will have a smaller cube in front of them that can be controlled using hand gestures. Both cubes will be scrambled to the same configuration so that it is easier for the user to follow.

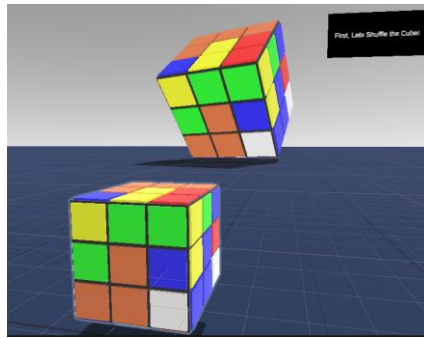


Figure 34: Cubes scrambled to the same configuration

The tutorial is divided into 8 sections namely, basic notations, white cross, first layer, second layer, yellow cross, last layer edges, orientate last layer corners, permute last layer corners. For this implementation, the tutorial is only taught up to the second layer as each section follows the same design and concept, and teaching the entire cube would take up too much time and is not feasible for testing.

For each of the section, the bigger cube will first demonstrate how to perform some of the moves. The algorithms used will be displayed on the screen for the user to follow. Once the user has achieved the same state as the bigger cube, the user can make a hand gesture to proceed to the next scene. Similarly, if the user did not fully catch the tutorial, another hand gesture can be used to restart the current scene.

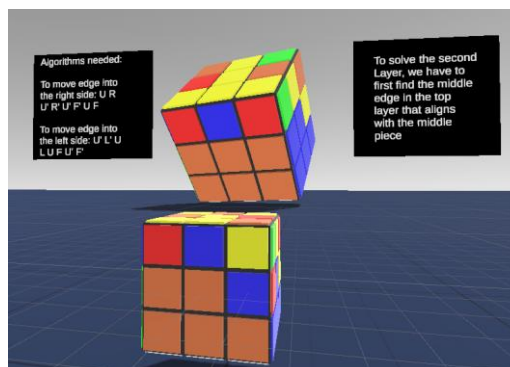


Figure 35: Learning to solve the second layer

3.5 Image Cube

The 'Image Cube' is an innovation added as a practice stage for users who have completed the tutorial to test their understanding of the solution. The image cube is slightly harder than the original Rubik's cube as there are no colours to guide the user and they have to rely on their ability to recognise the differences in the images. Another consideration is that the orientation of the centre pieces must be correct which is usually not a concern when solving the original Rubik's cube. Nevertheless, the 'Image Cube' can still be solved intuitively without the need to learn extra algorithms.

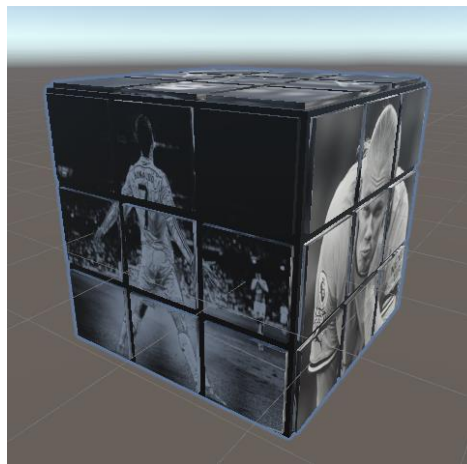


Figure 36: Image Cube

The scene design for the image cube is similar to the tutorial except there is no big cube to guide the user. One possible addition for the future would be to add a timer to track of the speed of the user's solves.

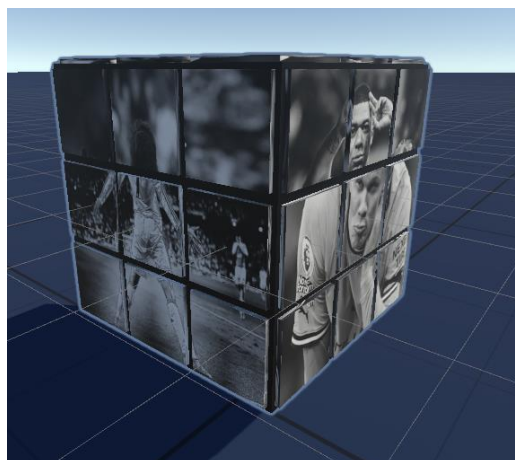


Figure 37: Solving the Image Cube

3.6 Hand Gestures

After adding the different functionalities and creating the scenes required, the last step is to allow the user to control the scenes and objects using hand gestures. While using hand gestures removes the need for controllers and makes the experience more seamless, the overuse of it may backfire as having too many hand gestures could confuse the user, take more time to familiarise, and affect the accuracy of recognition.

3.6.1 Setting Up

To use hand gestures in our tutorial, we have to first download and install the Oculus Integration Package from the Unity Asset Store.

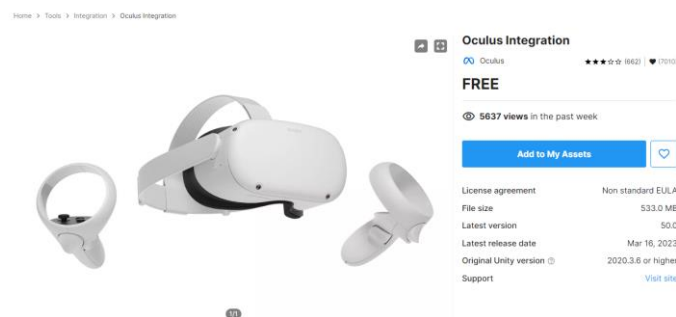


Figure 38: Oculus Integration Package

We can then add the “OVRCameraRig” asset into our scene and change the “Hand Tracking Support” to “Hands Only” with the frequency set to “High”. Next, we can add the “OVRHandPrefab” under our “LeftHandAnchor” and “RightHandAnchor” and ensure that the hand, skeleton and mesh type are all selected for the correct hands. This completes the basic setup, and we should be able to see our hands in the scene when plugged into our Oculus Quest 2.

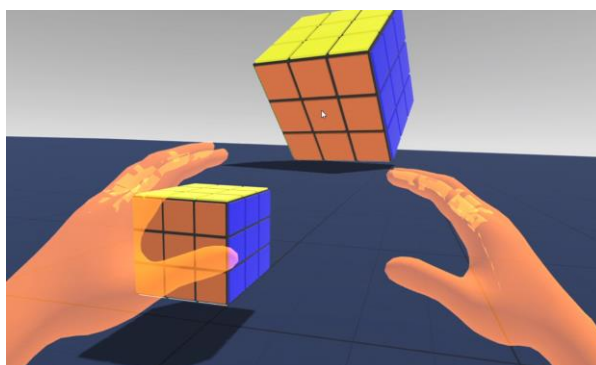


Figure 39: Hands visible in Unity Game View

3.6.2 Recording Hand Gestures

As it will take a very long time to manually input the finger data to create a hand gesture, we wrote a simple script to help us record our gestures. We first created a new structure for a gesture which consist of the name, the list of finger data in Vector3 and a Unity Event to trigger our desired function when the gesture is recognised.

```
[System.Serializable]
8 references
public struct Gesture
{
    public string name;
    public List<Vector3> fingerDatas;
    public UnityEvent onRecognized;
}
```

Figure 40: Creating gesture structure

From there we can create a save function which will trigger with a key press, to save the transform of each bone in the fingers into our new gesture.

```
void Save()
{
    Gesture g = new Gesture();
    g.name = "New Gesture";
    List<Vector3> data = new List<Vector3>();

    foreach (var bone in fingerbones)
    {
        data.Add(skeleton.transform.InverseTransformPoint(bone.Transform.position));
    }

    g.fingerDatas = data;
    gestures.Add(g);
}
```

Figure 41: Function to save gestures

Finally, we can go into play mode and begin creating and saving new gestures. We will have to copy the components before exiting play mode as all the changes we make during play mode will be deleted.

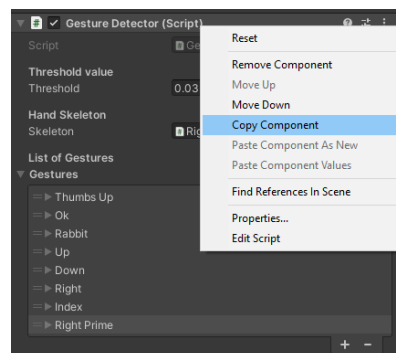


Figure 42: Copying gestures created

3.6.3 Recognising Hand Gestures

To recognise the gestures, we can iterate through the list of gestures and compare the finger data of the current finger positions to the ones we have saved and compute the difference between them. To reduce the amount of processing, we can set a threshold so that we can skip the gesture whenever the difference of any finger data is greater than the threshold.

If the difference is below the threshold for all the fingers, we can sum the differences and save this value and gesture as the one that is recognised for now. From there, we will continue to iterate and the gesture with the minimum difference in the finger data will be recognised as the one the user has made.

The threshold value is very important as a value that is too high will result in no gestures being detected whereas a value that is too low will incorrectly detect a gesture when it has not been made. Upon testing different values, I found out that 0.03 was the most accurate threshold for the gestures that I have created.

```
Gesture Recognize()
{
    Gesture currentGesture = new Gesture();
    float currentMin = Mathf.Infinity;

    foreach (var gesture in gestures)
    {
        float sumDistance = 0;

        bool isDiscarded = false;

        for (int i = 0; i < fingerbones.Count; i++)
        {
            Vector3 currentData = skeleton.transform.InverseTransformPoint(fingerbones[i].Transform.position);
            float distance = Vector3.Distance(currentData, gesture.fingerDatas[i]);

            if (distance > threshold)
            {
                isDiscarded = true;
                break;
            }

            sumDistance += distance;
        }

        if (!isDiscarded && sumDistance < currentMin)
        {
            currentMin = sumDistance;

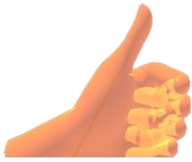
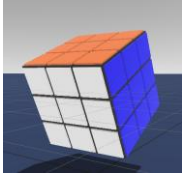
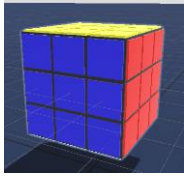


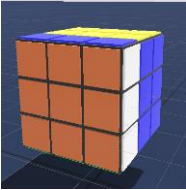
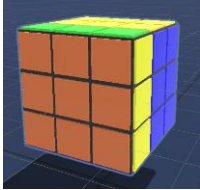



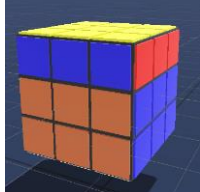



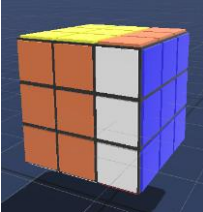



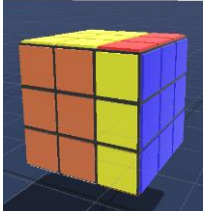

            currentGesture = gesture;
        }
    }

    return currentGesture;
}
```

Figure 43: Recognising Gestures

3.6.4 Mapping the Gestures to the Functions

The last part of the implementation is the map the gestures that we have created to trigger the different functionalities of the cube. In total, we have 7 gestures for each hand summarised in Table 2 below:

Left Hand	Left Output	Right Output	Right Hand
	Rotate Cube Up 	Rotate Cube Left 	
	F' 	F 	
	U' 	U 	
	L' 	R 	
	L 	R' 	


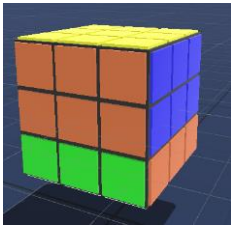
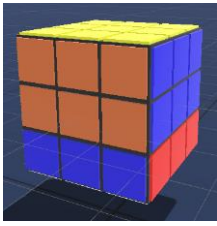



	<div>D</div> 	<div>D'</div> 	
	Reset Scene	Next Scene	

Table 2: Mapping of Gestures

3.7 Adding Audio

To add to the immersive experience, I added a simple background music as well as a clicking sound whenever a layer rotates. The timing of the sound has to be synchronised with the speed of rotation to make it sound natural.

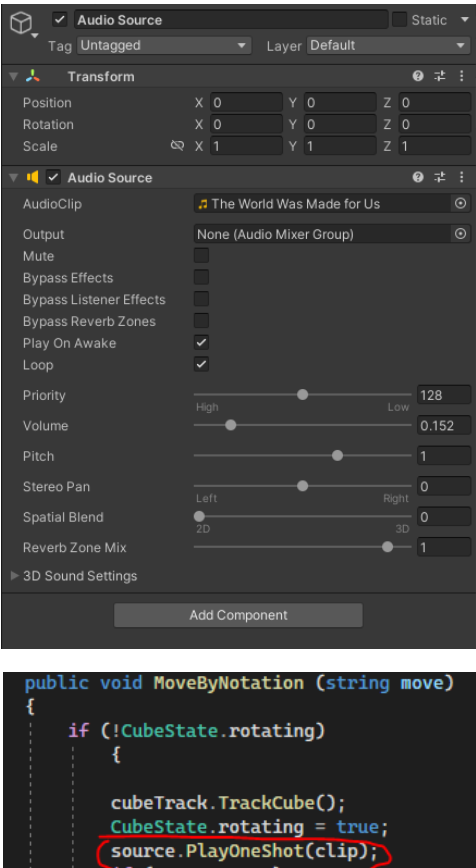


Figure 44: Adding click sound when layer rotates

Chapter 4 Evaluation

In this chapter, I will discuss some of the key findings and user feedback received, as well as the limitations and areas for improvement for my application.

4.1 Beta Testing

Due to time constraints and the difficulty in bringing my VR headset around, I was unable to conduct a proper beta testing of my application and only managed to get a few of my peers to try it. Nevertheless, I thought it would be beneficial to include some of their feedback in the report.

4.1.1 List of Participants

The list of participants who tested my application and their background is summarised in Table 3 below. The actual names of the participants are not listed for personal data protection.

No.	Age	Gender	Prior VR Experience	Able to Solve the Rubik's Cube?
1	24	M	No	Yes
2	24	M	No	Yes
3	25	M	Yes	No
4	22	F	No	No
5	24	M	Yes	No

Table 3: List of Participants

4.1.2 Questions

The list of questions posed to the participants is tabulated in Table 4 below. For questions 3a, 3b, and 8, participants were provided with a physical Rubik's Cube to test their ability.

No.	Question
Prior to testing the application	
1	Do you have experience in Virtual Reality and if so, could you describe the experience?
2	Have you tried playing the Rubik's Cube before?
3	Can you solve the Rubik's Cube?

3a	If you can't completely solve the Rubik's Cube, what is the furthest you can solve? (i.e., cross, one layer, two layers)
3b	If you can solve a Rubik's Cube, how fast can you do it and what is the method you are using?
4	Before trying the application, do you think using Virtual Reality can help you learn the Rubik's Cube more effectively than in real life?
After testing the application	
5	What is your opinion on the application?
6	Can you tell me whether there was anything that was good about the application (if any) and what are the things can be improved?
7	Do you have any other suggestions or features that you want to see in the application?
8	Now that you have tried it in VR, do you think it was effective in helping you learn the Rubik's Cube?
9	*For participants who are unable to solve a Rubik's Cube* Can you try to solve the cube again and see how far you can go?

Table 4: List of Questions

4.2 Results and Responses

The full open-ended responses can be found in the Appendix. For the purpose of this report, I will summarise some of the responses and common feedback provided by the participants.

4.2.1 Overall Opinion

Prior to testing the application, all the participants did not believe that learning the Rubik's Cube in VR could be more effective than in real life. After testing the application, the general sentiment remained the same, but this was limited by my flawed implementation of the application. Most of the participants felt that there was the potential for this to be an effective learning tool if the controls were more intuitive, and the tutorial was better designed. The hand gestures appealed to all the participants, but most agreed that the accuracy could be improved.

4.2.2 Suggestions for Improvement

The improvement that most participants would like to see was the ability to grab and move the cube with their hands, rather than controlling it far away with gestures. One of the participants mentioned that this would be a “gamechanger” if it was implemented.

Another suggestion shared by the participants was to implement the other sized cubes (4x4, 5x5) as they are not as accessible as the original 3x3 Rubik’s Cube. The image cube was too challenging for most of the participants, even for those who are able to solve the Rubik’s cube. One of the participant suggested to use an image cube with colours (instead of black and white), or even a Supercube which is a Rubik’s Cube with arrows, where the objective is to solve the cube with all the arrows of the same colour pointing in the same direction.



Figure 45: Example of a Supercube

One very interesting suggestion shared by the participant with the most experience in Rubik’s Cube was to use the application as a training tool to memorise algorithms. For more advanced methods such as the Cross, First 2 Layers, Orientation, Permutation (CFOP) method, there is a total of 78 algorithms to memorise to solve the last layer of the cube, 57 to orientate the last layer (OLL), and 21 to permute the last layer (PLL). It is troublesome to set up the cube to be in the exact scenario to practise the algorithm in real life, and this is where learning using VR may come in handy as we are able to shuffle and reset the cube to the scenario we want easily. When starting to learn the algorithms, it is common to forget some of them, and it is a hassle to always refer online and search for the algorithm for the exact pattern. Learning in VR can help to overcome this by recognising the pattern and providing the algorithm required when the user requests for help.

Other suggestions include ability to control the speed of the rotation, adding voice to the tutorial, highlighting the pieces in the tutorial so that it is easier to follow, and beautify the scene by adding more things.

4.2.3 Interesting Observations

An interesting observation made was that there were participants who rotated the cube differently after testing out the application. Prior to using the application, two of the participants rotated layers of the physical cube either using multiple fingers or their entire hand, but they were comfortable with only using one finger on their second attempt. This meant that using hand gestures has an impact on the user's muscle memory and they may interact with the object differently in real life. It is hence possible that the combination of hand tracking and VR may be extended to other use cases where finger positions are important such as teaching children how to properly hold utensils, or to play different chords on the guitar.

Another interesting observation was that none of the participants complained about VR sickness even though it was their first time experiencing it for some of them. This may be because the testing itself was only for a short duration of less than 10 minutes and the gameplay itself is stationary and does not involve any movement.

4.3 Limitations

One of the biggest limitations was that the scope of the beta testing was extremely small with only 5 participants, and we are unable to draw any conclusive results based on it. The participants were also from the same age group and, it would have been more useful if we were able to test it out over a wider age range, from adolescence to the elderly to gain their perspective on learning using VR.

For the application itself, the main limitations stemmed from the difficulty in implementing a fully functional, intuitive tutorial by myself. The project underwent many changes along the way, as there were many obstacles and features that were unable to be implemented due to my lack of knowledge and ability.

4.4 Key Findings

After gathering all the user feedback and from my own experience throughout this project, these are the key findings I have gathered regarding Virtual Reality and Immersive e-Learning:

4.4.1 Adequate Substitute but never the Real Thing

Virtual Reality has come a long way and is arguably an adequate substitute for learning and picking up new skills such as the Rubik's Cube, piano, guitar, racket sports etc., but the level of immersion is still unable to replicate the real-world experience. This is because the current immersion of VR still relies heavily on the user's sight and hearing, and the aspects of taste, smell and most importantly touch, is still not integrated into the experience.

While the advancement of VR technology such as VR haptic gloves is slowly bridging the gap by providing a sense of touch in the virtual environment, they are still limited to only the hands of the user and is unable to replicate the full body experience.

The strength in VR comes when the simulation possesses risk or danger in the actual world such as roller coaster rides, medical training or flight simulations, or when the objects or environment required are not as easily accessible e.g., a PC building simulator or playing a musical instrument in a famous concert hall. The benefits of VR in learning, however, starts to fade in a perfectly safe environment such as learning to solve a Rubik's Cube.

4.4.2 Balancing Costs and Rewards

The quality of experience comes with the quality of the application which in turn comes with cost. The efficiency of learning using VR is largely dependent on the implementation of the application, and it has to be of a certain minimum standard depending on the scenario, in order to achieve any positive results.

Implementing VR technology in e-Learning comes with significant costs in terms of hardware, software, and even health. Prior to that, it is important to carefully consider the costs involved and weigh them against the benefits it can provide.

Ignoring the lifespan of the headset itself, one major benefit of implementing a VR solution is its long lifespan and consistency it provides. Landmarks, museums, and theme parks may be demolished, musical instruments may break, teachers may change; but the virtual tours and roller coaster rides, virtual piano and guitar, and the learning content will always be there once it has been created.

Organisations should always evaluate the long-term costs and rewards of the solution and strike a balance to decide whether the implementation is worth the cost.

4.4.3 Customisability and Scalability are Key

The greatest advantages to using VR and immersive e-learning lies in its customisability and scalability. Customisation can include everything from the design of the learning environment to the pace and difficulty of the content, and even to the content itself. VR can be tailored to create any possible and even impossible scenarios and the most amazing part is that it is able to switch from one scenario to another in a matter of seconds. This is where immersive learning in VR may have an advantage over traditional learning when the setup of the scenario requires significant time and resources.

Scalability is another major factor that makes VR a better option compared to traditional learning. A physical Rubik's Cube can only be used by one individual at a time, but a virtual cube can be used by millions at the same time. This may appear insignificant as a Rubik's Cube is easily accessible, but the significance scales when the resource in question is more expensive or restricted such as a medical operating theatre. Similarly, people around the world can ride on the same virtual roller coaster without having the need to queue.

Ultimately, I believe the question is no longer whether VR and Immersive e-Learning works, and more of whether the situation calls for it and whether the implementation is worth the cost.

Chapter 5 Conclusion

5.1 Conclusion

In conclusion, although the current implementation of a Rubik's Cube tutorial in VR does not seem to be more effective than Traditional Learning in real life, I believe that there are very specific areas where VR has an advantage in terms of its ability to customise and scale.

However, it is imperative that we first minimally create an experience and interactions that is close to reality before we can start to exploit the advantages and edge that VR has to offer.

5.2 Recommendations for Future Work

I feel that future work should be focused on improving the interactions between hand tracking and the virtual environment. My current implementation only makes use of hand gestures which is essentially the location of the fingers relative to the hand without any consideration for the environment or the objects in it. By combining hand gestures and the position of the hands relative to the objects or environment, I believe it can create a lot more possibilities (e.g., pointing at different objects to trigger different reactions) and make the experience more intuitive.

With regards to the application itself, some possible areas for future work can be to teach more advanced algorithms, expanding the interaction beyond 3x3 Rubik's Cube and into other variations, and exploring the possibility of setting the cube to a particular state for users to practise and memorise algorithms.

References

- [1] Turi, J. (2014, February 16). *The sights and scents of the Sensorama Simulator*. Engadget. Retrieved from <https://www.engadget.com/2014-02-16-morton-heiligs-sensorama-simulator.html>
- [2] P&S Intelligence. (2022). *Extended Reality Market Research Report*. <https://www.psmarketresearch.com/market-analysis/extended-reality-xr-market-insights>
- [3] Perkins Coie. (2019). *2019 Augmented and Virtual Reality Survey Report*. <https://www.perkinscoie.com/images/content/2/1/v4/218679/2019-VR-AR-Survey-Digital-v1.pdf>
- [4] Cureton, D. (2023, February 28). *Virtual reality statistics to know in 2023*. XR Today. Retrieved from <https://www.xrtoday.com/virtual-reality/virtual-reality-statistics-to-know-in-2023/>
- [5] Iberdrola. (2018). *Virtual reality: Another world within sight*. Retrieved from <https://www.iberdrola.com/innovation/virtual-reality>
- [6] Loeffler, J. (2021, September 30). *The history and science of virtual reality headsets*. Interesting Engineering. Retrieved from <https://interestingengineering.com/innovation/the-history-and-science-of-virtual-reality-headsets>
- [7] Mishra, Preetam. (2021). *“a step toward future” – VR*. Retrieved from https://www.researchgate.net/publication/354065442_VR-Research_paper
- [8] Aniwaa. (2021, August 6). *Types of VR headsets: PC VR, standalone VR, Smartphone VR*. Retrieved from <https://www.aniwaa.com/guide/vr-ar/types-of-vr-headsets/>
- [9] National University of Singapore. (2020, August 11). *Not playing games: VR for medical training*. Retrieved from <https://news.nus.edu.sg/not-playing-games-vr-for-medical-training/>
- [10] Hall, S. A. (2022, August 8). *Music in the metaverse - learn to play the piano in this new Augmented Reality App*. Classic FM. Retrieved from <https://www.classicfm.com/discover-music/instruments/piano/metaverse-virtual-augmented-reality/>

- [11] Gera, R. R. S. (2019, August 13). *Making learning more accessible for the differently abled with VR*. Emerging Education Technologies. Retrieved from <https://www.emergingedtech.com/2019/08/making-learning-more-accessible-for-differently-abled-virtual-reality/>
- [12] Gera, E. (2018, December 11). *How VR is being used to help children with learning disabilities, autism*. Variety. Retrieved from <https://variety.com/2018/digital/features/voiss-interview-vr-hmd-1203086576/>
- [13] Storm, A. (2023, February 28). *What is immersive learning?* Thinkific. Retrieved from <https://www.thinkific.com/blog/immersive-learning/>
- [14] Mark. (2022, June 23). *What is virtual reality hand tracking? - VRX by VR expert*. VRX. Retrieved from <https://vr.x.vr-expert.com/what-is-virtual-reality-hand-tracking/>
- [15] Butler, S. (2021, March 10). *How oculus quest hand tracking technology works*. Online Tech Tips. Retrieved from <https://www.online-tech-tips.com/gaming/how-oculus-quest-hand-tracking-technology-works/>
- [16] Baker, H. (2022, July 4). *Top 10 best quest 2 hand tracking games & apps - summer 2022*. UploadVR. Retrieved from <https://uploadvr.com/best-oculus-quest-hand-tracking/>
- [17] SenseGlove. (2022, November 3). *VR gloves and how they work*. SenseGlove. Retrieved from <https://www.senseglove.com/a-guide-to-vr-gloves/>
- [18] Ochanji, S. (2020, August 23). *This new haptic VR glove provides a realistic warm/cold feeling*. Virtual Reality Times. Retrieved from <https://virtualrealitytimes.com/2020/08/23/this-new-haptic-vr-glove-provides-a-realistic-warm-cold-feeling/>
- [19] Barto, A. (2022, March 14). *Cost of custom virtual reality training: Full VR price, cost factors, and benefits [2022]*. Roundtable Learning. Retrieved from <https://roundtablelearning.com/cost-custom-virtual-reality-training-full-vr-price-cost-factors-benefits-2022/#:~:text=So%2C%20there%20you%20have%20it,scale%2C%20these%20prices%20will%20vary.>

- [20] L., S. (2022, December 22). *How much does it cost to develop a VR app in details*. Cleveroad Inc. - Web and App development company. Retrieved from <https://www.cleveroad.com/blog/how-much-does-it-cost-to-develop-a-vr-app--discover-in-market-research/>
- [21] Anses. (2022, June 29). *What are the risks of virtual reality and augmented reality, and what good practices does anses recommend?* Retrieved from <https://www.anses.fr/en/content/what-are-risks-virtual-reality-and-augmented-reality-and-what-good-practices-does-anses#:~:text=Exposure%20to%20virtual%20reality%20can,first%20few%20minutes%20of%20use.>
- [22] LaMotte, S. (2017, December 13). *The very real health dangers of virtual reality*. CNN. Retrieved from <https://edition.cnn.com/2017/12/13/health/virtual-reality-vr-dangers-safety/index.html>
- [23] Kim, M. (2015, February 18). *The good and the bad of escaping to virtual reality*. The Atlantic. Retrieved from <https://www.theatlantic.com/health/archive/2015/02/the-good-and-the-bad-of-escaping-to-virtual-reality/385134/>
- [24] Bem, N. N. (2021, January 10). *5 benefits of learning how to solve a Rubik's Cube*. Goodnet. Retrieved from <https://www.goodnet.org/articles/5-benefits-learning-how-to-solve-rubiks-cube>
- [25] Koebler, J. (2011, October 27). *Rubik's Cube finds a home in American classrooms*. US News. Retrieved from <https://www.usnews.com/news/blogs/stem-education/2011/10/27/rubiks-cube-finds-a-home-in-american-classrooms>
- [26] Ruwix. (n.d.). *Rubik's cube notation - What the rotation letters mean: F R' U2*. Retrieved from <https://ruwix.com/the-rubiks-cube/notation/#:~:text=For%20the%20beginner's%20method%20you,or%20reorient%20the%20whole%20cube.>
- [27] Heyes, D. (2023, January 4). *Sci-Fi Bedroom*. Sketchfab. Retrieved from <https://sketchfab.com/3d-models/sci-fi-bedroom-9bf95b0af80244d693c22d160e5c5405>

Appendix

Responses from Participants in Testing

*To note: Questions 3a, 3b and 9 include observations made by me in addition to the response provided by the participant.

Participant 1	
1	No.
2	Yes, I started in primary 3 or primary 4 but I stopped playing for a while now.
3	Yes.
3a	NIL.
3b	I use the CFOP method, but I use the 2-Look OLL instead of the full OLL. *Participant solved the cube in 32 seconds.
4	No, I think it is important to have a physical feel of the cube.
5	I think it was quite interesting because I never tried VR before. The hand-tracking feature was quite cool but there were times when it wasn't very responsive or did the wrong move.
6	I thought the image cube was quite fun but it was a bit hard to see because the cube was quite far away. For improvement, I think maybe the speed of the rotation was too slow for me and I don't think someone who is trying to speed cube will use the application. Some of the gestures wasn't very intuitive especially the F and F prime moves.
7	I feel that the application can be useful for memorising algorithms if I can automatically set it to a certain scenario. Let's say if I want to practice OLL, maybe you can help me shuffle the cube to one of the 50+ scenarios then once I solve it, you can change to another scenario.
8	Not for me, I think a physical cube is always better.
9	NIL

Participant 2	
1	No.
2	Yes, I think I first played around Primary 6, but I stopped after that.
3	Yes, I hope I can still remember.
3a	NIL.
3b	I don't know what this method is called. * Participant solved the cube in 2 minutes 24 second using the Beginner's method. *
4	I don't see how it can be more effective.
5	The tutorial was boring but maybe that's because I already know how to solve it. The image cube was way too hard, I doubt anyone who just learned the cube can solve it. Maybe can try using images with colours or a supercube instead.
6	The hand tracking was interesting, but the accuracy can definitely be improved. Will also be good if I can move the cube by touching it instead of controlling it from far away.
7	I wanted to try the 4x4 or 5x5 cubes but sadly they were not implemented.
8	I can see how it be helpful for a beginner, but I think you can achieve the same results from watching a YouTube tutorial. The other sized cubes would have been more effective because I don't own those.
9	NIL

Participant 3	
1	Yes. I tried it at my friend's house once. I played a rhythm game where you had to hit the notes as they came towards you.
2	Yes, I bought one when I was in primary school but never learned to solve it.
3	No.
3a	* Participant solved one layer, but the edges and corners were misaligned*
3b	NIL
4	Not really, I don't think so.
5	The tutorial can be better designed. It was quite boring to just read the words on the screen so maybe adding voice will help. It will also be nice to highlight the pieces the tutorial is trying to refer to because it is quite hard to follow sometimes.
6	I thought the controls were well thought out. It was simple to remember and feels intuitive.
7	No.
8	Yes, I learned a new method to solve the first layer.
9	* Participant was able to solve one layer with the edges and corners in the right positions. An interesting observation was that the participant was using 2 fingers to turn the layers before testing the application, but changed to using 1 finger for some of the layers after using the application*

Participant 4	
1	Nope but I'm excited to try it.
2	Try as in play with my friend's cube for like 5 minutes then yes.
3	No.
3a	* Participant gave up after a while*
3b	NIL
4	I don't know, it could be?
5	I think the Rubik's cube is too hard for me to understand but it was quite fun. Using my hands to control the game was also very cool.
6	The first room was very nice but the tutorial was quite bland, maybe can add more stuff to fill up the space.
7	Maybe can move the cube nearer and let the user rotate the cube with their hands instead. It's difficult to see the entire cube when you can only rotate it by 90 degrees.
8	I guess that the fact that it was in VR helped me pay attention to something that I was not interested in, but the effect might disappear if I use it for a long time.
9	* Participant was able to solve the cross but forgot the algorithm for the edges. Similar observation was made where participant was moving the layers with all 5 fingers previously, but changed to using just one or two fingers*

Participant 5	
1	I played it once before at a place called Sandbox VR. It was a shooting game where you had to kill zombies and stuff.
2	Yes, when I was young.
3	No, I only managed to solve it once when my friend helped me, but I can't remember anymore.
3a	* Participant solved one layer, but the edges and corners were misaligned*
3b	NIL
4	I feel like there is potential for that, but that would require a very good design and implementation.
5	The hand gestures were fun, would be nice if I could try the other cubes too.
6	Controlling the cube with my hands was quite smooth, but there were delays when I tried to swap to another move, and it took awhile to recognise the gesture.
7	Maybe let the user grab the cube and rotate it by touching instead of making a gesture. I think that will be a gamechanger.
8	With a better implementation, I'm sure it could but not for now.
9	* Participant was able to solve one layer with the edges and corners in the right positions.