한국컴퓨터그래픽스학회
Korea Computer Graphics Society Vol. 30, No. 3, P. 43~49

XR Hands를 통한 가상 객체들과의 상호 작용에 관한 연구

조범준 김성기O*

상명대학교 게임학과, 조선대학교 컴퓨터공학과

202131045@sangmyung.kr, skkim@chosun.ac.kr

A Study on the Interaction with Virtual Objects through XR Hands

BeomJun Jo SeongKi KimO*

Department of Game Design and Development, Sangmyung University
Department of Computer Engineering, Chosun University

요 약

핸드트래킹이 주된 조작이 되는 기기가 출시됨에 따라 현재 확장현실 분야의 관심사 중 하나는 핸드트래킹이다. 핸드트래킹은 사용자에게 몰입감 및 현실감 측면에서 장점이 있으며 그에 따라 교육, 엔터테인먼트, 의료 등 다양한 분야에서 활용되고 있다. 활쏘는 동작은 양손을 동시에 써야하는 동시에 표적을 맞추기 위해서는 정교함을 요하는 과거 문화적 스포츠적 의의를 가지는 동작이다. 본 연구는 이러한 활쏘는 동작을 구현하는 것을 목표로 했다. 따라서 본 논문은 유니티가 제공하는 XR Hands 패키지를 활용하여 손동작을 인식하도록 했으며 기반이 되는 OpenXR 에 대한 탐구했다. 최종적으로는 활 쏘는 동작을 구현했고 메타 퀘스트 2 에서 테스트했다.

Abstract

Hand tracking is currently one of the most promising technologies in XR with the release of extended reality (XR) devices, in which hand tracking is used as the main manipulation. Hand tracking offers advantages in terms of immersion and realism, and as a result, it is being employed in a range of fields, including education, business, and medical care. The archery movement requires using both hands at the same time, but requires sophistication to hit the target and is a movement that has cultural and sports significance in the past. This study aimed to implement this archery movement. Therefore, this paper used the XR Hands package provided by Unity to recognize hand movements, explored the underlying OpenXR, and finally implemented the archery movement and tested it in Meta Quest 2.

키워드: 가상손 , 가상현실, 손동작, 유니티

Keywords: Hand Gesture, Unity, Virtual Reality, XR Hands

1. Introduction

Currently, a plenty of head-mounted displays (HMDs) for virtual reality (VR) are available on the market. With the advancement of technology, commercial HMDs are not limited to VR, but also implement extended reality (XR) beyond augmented reality (AR). These devices have their own controllers, but they also support hand tracking, enabling users to manipulate devices without using the controller. Furthermore, devices where the primary method of interaction is hand tracking have also been released, such as Apple Vision Pro [1].

Hand tracking technology enhances the sense of immersion and reality by facilitating natural interaction between users and virtual environments. Moreover, the absence of a controller offers tangible benefits, including the convenience of not having to use one. These technological advances facilitate the development of innovative applications in a range of fields, including education, entertainment, and medical care.

In this study, an archery motion based on hand tracking technology was implemented using the Unity engine. Archery has been a significant pastime in various cultures and sports, spanning from ancient times to the present. It necessitates intricate hand movements and control.

The objective of this study is to develop a system that enables users to perform a series of actions, including pulling, aiming, and firing a bow, in a virtual environment through hand tracking. The objective of this study is to investigate the potential for intuitive and immersive interaction in virtual reality. Furthermore, the study aims to demonstrate the feasibility of extending this approach to a range of applications in the future.

*corresponding author: SeongKi Kim/Chosun University(skkim@chosun.ac.kr)

Received : 2024.06.15./ Review completed : 1st 2024.07.02. / Accepted : 2024.07.05.
DOI : 10.15701/kcgs.2024.30.3.43
ISSN : 1975-7883(Print)/2383-529X(Online)

The remainder of this article is structured as follows: Section 2 presents related works on hand tracking using OpenXR. Section 3 outlines the implementation of the system. Section 4 provides a conclusion to this paper.

2. Related Works

This section describes the research using the underlying technology OpenXR, and the Unity's XR hand used in this paper.

2.1 OpenXR

Prior to the advent of OpenXR [2], there was OpenVR [3], developed by Valve Corporation for use with Steam. OpenVR was an API for SteamVR, which facilitated seamless integration with platforms that supported SteamVR. It is noteworthy that OpenVR continues to support the latest versions of Unreal and Unity through updates. However, OpenXR has emerged as the dominant API, becoming the norm in the industry.

OpenXR is a standard API developed by the Khronos Group that provides compatibility for XR devices, thereby facilitating the creation of a consistent development environment as shown in Figure 1. OpenXR supports a range of advanced interaction technologies, including HMDs, controllers, and tracking, including hand tracking and eye tracking. This enables developers to create a consistent user experience across various platforms.

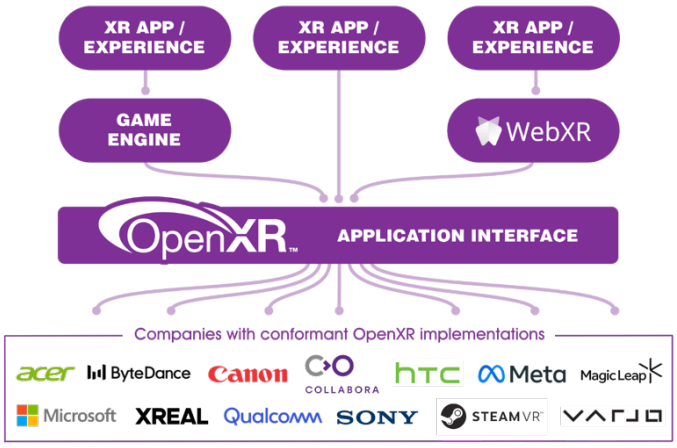


Figure 1. the role of OpenXR [2]

OpenXR is still in use and many studies are based on it. S.Li [4] has implemented a chemical laboratory in a virtual environment. Users can safely experience the experiment firsthand. M.Huzaifa et al. [5] present the Illinois Extended Reality Testbed (ILLIXR). It uses state of the art XR components to provide detailed data on performance, power and quality of experience for OpenXR based systems. C.Runde [6] says that standardization can make it easier to switch to other tools, increasing work efficiency and reducing costs for companies. WebXR [7] also aims at standardization, but unlike OpenXR, it uses WebGL, Three.js, etc. and is mainly web based [8]. However, OpenXR supports a wider variety of hardware.

These aspects show that OpenXR is robust.

2.2 XR Hands

In Unity, several packages are available for the development of virtual reality (VR) applications, including several toolkits such as VRTK and MRTK [9]. One such package is XR Hand, which provides functionality for the development of VR hands. The XR Hands package enables users to access the data obtained during hand tracking. The XRHandSubsystem is employed to import data, which utilizes the OpenXR plugin. In other words, the XR Hands package is based on OpenXR.

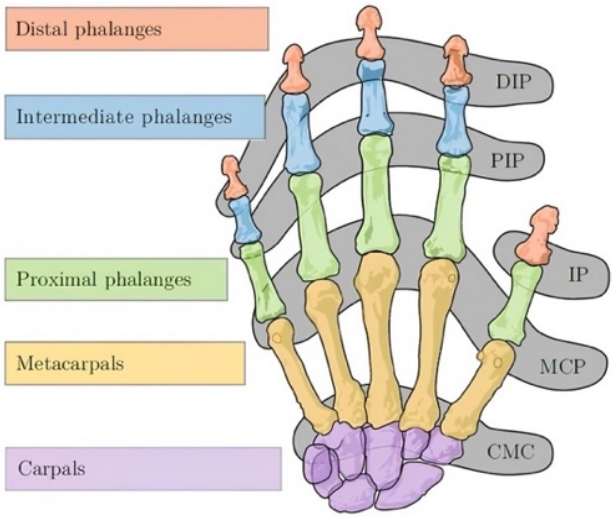


Figure 2. joints of real hand [11]

As illustrated in Figure 2, there are 19 joints in a real person's hand. The thumb has an interphalangeal joint (IP), a metacarpophalangeal joint (MCP), a carpometacarpal joint (CMC), and the other four fingers have a distal interphalangeal joint (DIP), a proximal interphalangeal joint (PIP), an MCP, and a CMC [10]. Furthermore, OpenXR draws 26 points on the hand, representing the fingertips, wrist, and palm to determine the orientation of the hand.

Figure 3. hand points of OpenXR [12]

In Figure 3, OpenXR represents the hand with 26 points; 4 joints for the thumb, 5 joints for the others, palm and wrist. Because XR Hands is OpenXR-based, XR Hands also represents the hand with 26 points, as shown in Figure 4. But the difference is that the positive z direction is OpenXR is backward, while XR Hands is forward.

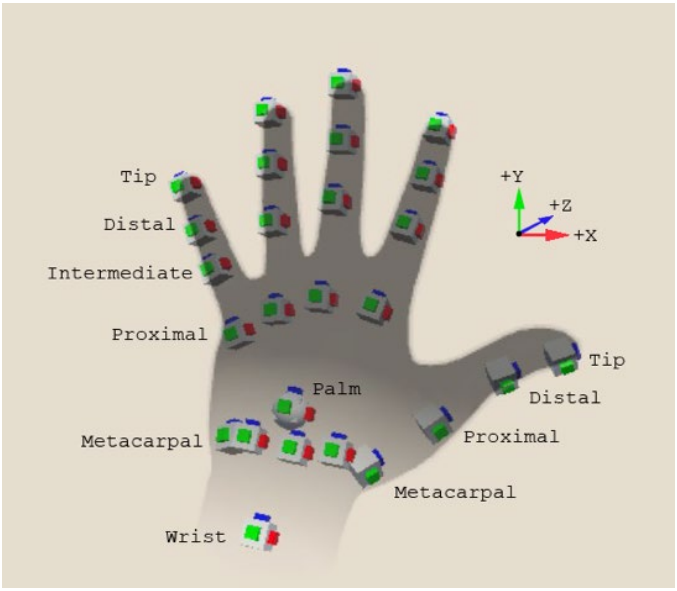


Figure 4. hand landmarks of XR Hands [13]

3. Implementation

3.1 XR Hands

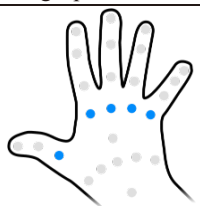
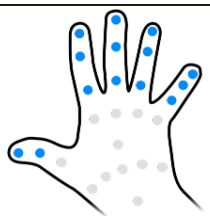
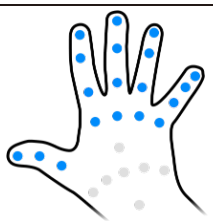
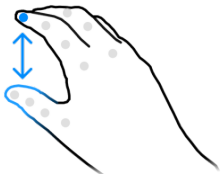
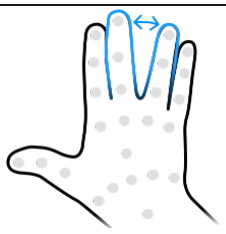
As stated in the previous section, XR Hands is an OpenXR-based API provided by Unity. Since version 1.4 released this year, gesture detection has been supported. The method of making gestures using this API is as follows;

1. Make XR hand shape object. And adjust the values for the bending of each finger to create desired hand shape.
2. Set the XR hand pose object by adding values for the hand's direction, orientation and other parameters. If a specific pose is not required, this step can be skipped.
3. Use a script to add events that will be triggered when the gesture is performed.

3.1.1 XR Hand Shape

XR Hands made it possible to save gestures in the form of XR Hand Shape. Each finger is assigned a total of five values; base curl, tip curl, full curl, pinch, and spread [14]. These values are normalized to a minimum of 0 and a maximum of 1.

Table 1. finger parts affected by each value

Base curl		MCP
Tip curl		The outer portion of finger
Full curl		The whole of finger
Pinch		fingertip
Spread		Between adjacent fingers

The base curl sees the bending between the finger and the palm. The focus is on the MCP in Figure 2, and it is 0 when the hand is straightened, and 1 when the finger and palm are perpendicular.

The tip curl considers the upper portions of the finger, encompassing the entire digit, and assesses the bending of the joint corresponding to the tip, distal, and intermediate regions, as illustrated in Figure 4. If the finger is not bent, the value is 0, and if the finger is bent maximally, when the fingertip is closest to the MCP, the value is 1.

The full curl specifies the overall bending of the finger. It is also 0 when all fingers are stretched out and 1 when folded completely. In other words, the full curl is the average value of the base curl and the tip curl.

The pinch represents the distance between the thumb and the other fingertip. It has a value of 0 when the finger is opened and a value of 1 when the other fingertip and thumb are fully joined. As it is based on the thumb, the thumb is not a value to have.

Finally, the spread is a value affected by the angle with the adjacent finger. The value is 0 if the angle between the side finger and the fingertip is parallel, and 1 if the fingertip is positioned at the farthest distance from the side finger.

As shown in Figure 5, a hand shape can be created by setting the desired value and the tolerance for each finger. For example, if you set the desired value of full curl for all fingers to 1, this means that all fingers are folded, so the hand shape represents a fist, as shown in Figure 6.

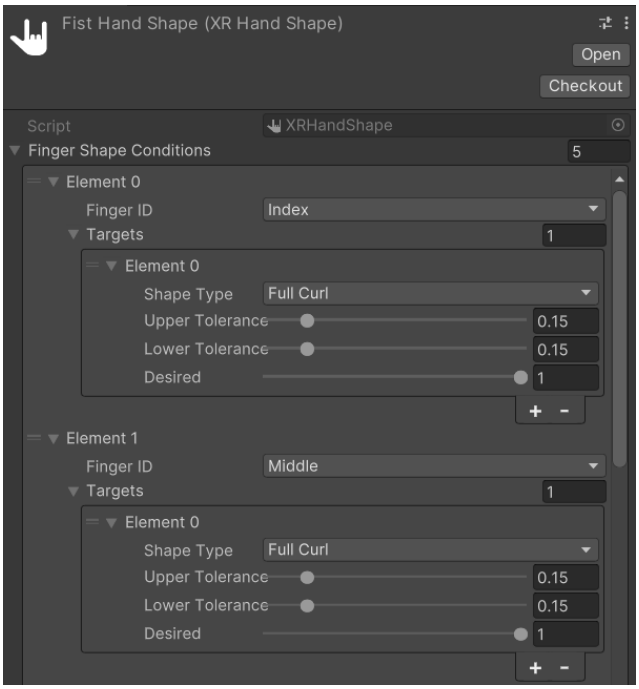


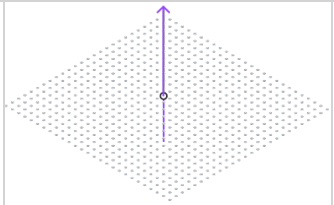
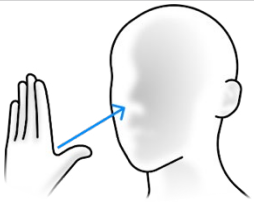
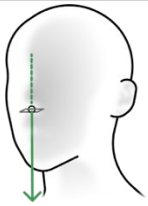
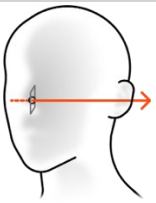
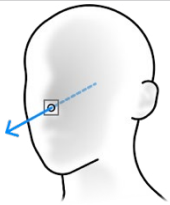
Figure 5. setting example of hand shape

Figure 6. the gesture
when the values of full curl for all fingers are 1

3.1.2 XR Hand Pose

XR Hand Pose is also a novel form offered to make gestures. To complete the gesture, the previously created hand shape is augmented with orientation and directional information. The process of creating the hand pose may be omitted if it is not necessary to set conditions such as a specific posture. The hand pose comprises three main conditions [15]; hand axis, alignment, reference direction.

Table 2. the type of the reference direction

Origin up		perpendicular to the ground
Hand to head		from the hand to the head
Chin direction		from the center of the head to the jaw
Ear direction		from the center of the head to the ear (different depending on the hand)
Nose direction		from the center of the head to the nose

First, there are three values of the hand axis; finger extended direction, thumb extended direction, palm direction. Finger extended direction is the positive Z direction in Fig. 4 in the direction from the palm to the finger, based entirely on the single hand.

The reference direction is the direction in which the gesture will be made compared to the hand axis. It has five directions as shown in Table 2; origin up, head to head, nose direction, chin direction, ear direction. Origin up is the positive Y axis of XR Origin. In user standards, it is the positive Y axis of the ground. Hand to head is the direction to the user's face. Nose direction is the direction from the center of the user's head to the nose. Chin direction is the direction from the center to the nose. Ear direction is the direction from the center to the ear. The left hand is directed to the left ear and the right hand is directed to the right.

Finally, Alignments define the relationship between hand axis and reference direction; aligns with, perpendicular to, opposite to. Aligns with defines whether two directions are the same, perceptual to is vertical, and opposite to is contrary to each other.

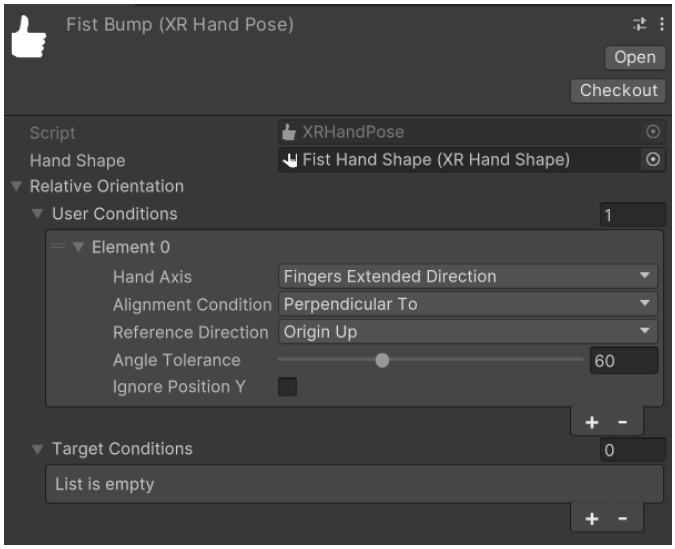


Figure 7. setting example of hand pose

The addition of a direction and tolerance to the hand pose, as illustrated in Figure 7, results in the formation of a hand pose. For example, if the hand is set to finger extended direction, origin up, and perpendicular to the fist shape, it becomes a fist bump gesture with the back of the hand facing upwards as shown in Figure 8.

Figure 8. fist bump gesture

3.2 Implementation

The grasping motion has been implemented in a rudimentary form even before the gesture was introduced. Consequently, if a XRGrabInteractable class is added to the object to be held, the object can be held. However, since the basic motion is implemented with a pinch performed by the thumb and index finger, there are instances where it cannot be recognized due to circumstances such as hand rotation.

In Figure 9, the bow was held using a fist gesture using the method in Section 3.1.1, and the bow was operated using a pinch gesture.

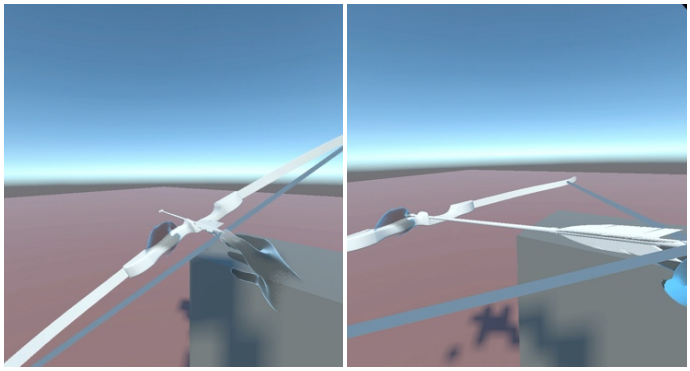


Figure 9. implementation of an archery motion in VR

3.3 Result

The XR Hands provides seven hand shapes as an example; fist bump, grab, open palm up, point at, shaka, thumb down, and thumb up. In theory, after making the hand pose and hand shape, the hand movement made can be recognized. However, in order to ensure that the hand shape and hand pose function correctly at the desired time, the desired value and error value must be identified by repeatedly performing the same action.

The HMD will continue to track the user's hand if it is recognized while the user is not using a controller. Then, the tracked hand is covered with a hand mesh, thereby allowing the user to continue to perceive their hand even in the absence of pass-through. Should gesture recognition have been implemented in this state, the user can utilize gestures. By implementing the gesture, the detection time was obtained as the difference in time after the gesture was judged before and after the gesture was judged, and Time.timeSinceLevelLoad was used. The computer running the editor is equipped with a Ryzen 7 4800H processor and a GTX 1660Ti as a GPU.

The detection time was calculated by averaging the time taken after performing 500 random gestures. The mean detection time was 24.8 microseconds, with a minimum of 15.3 microseconds and a maximum of 53.4 microseconds. Nevertheless, this detection time is not indicative of the accuracy. It is acceptable for the detector to identify a single motion. However, if it can detect multiple hand gestures, an intermediate level of hand gestures may emerge. To illustrate, when the system is instructed to detect a "grab" and a "fist," it identifies the "grab" shape while also detecting the "fist" immediately thereafter. Figure 10 below illustrates examples of grab and fist motion.

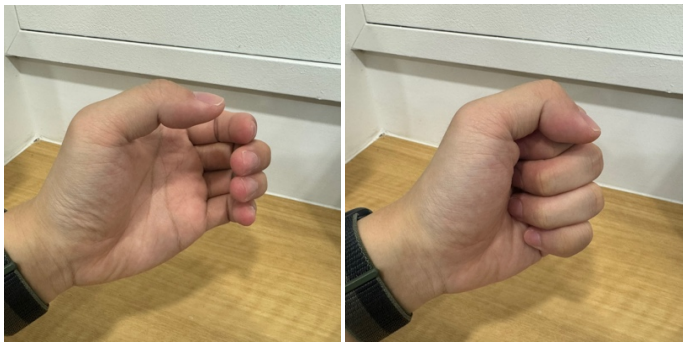


Figure 10. grab and fist gesture

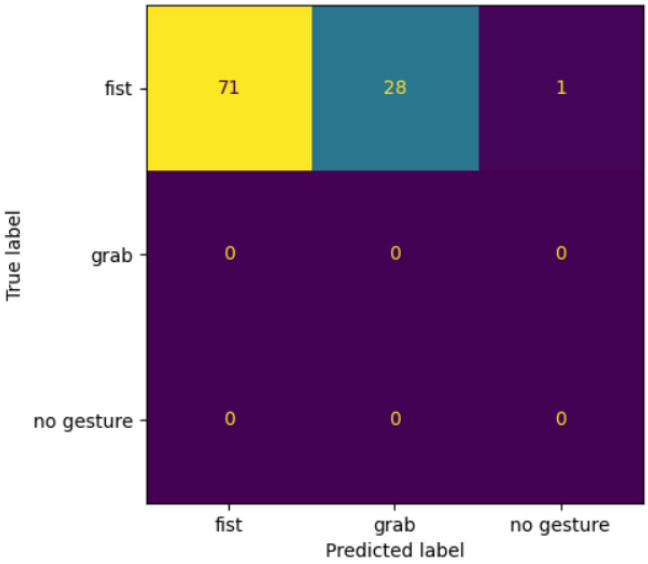


Figure 11. Confusion matrix of detection result of fist 100 times

Figure 11 shows the confusion matrix of the result of clenching fists 100 times when fist, grab could be detected. During the 100 repetitions of clenching fists, recognition was achieved 99 times. However, the intermediate stage motion "Grab" was recognized before fist motion was recognized 28 times. Consequently, if an intermediate stage hand motion has been implemented and the hand motions of both ends must be recognized with minimal delay, it is imperative to be cognizant of the intermediate stage.

And there was no difference in detection time, whether it was a fist shape containing five finger values or a fist bump with additional values in the shape. Regardless of the number of values in the hand shape and pose, the detection time took 20 microseconds as above.

Therefore, if users want the desired gesture to be detected correctly, it is necessary not to add an intermediate gesture or to keep the gesture short but constant.

Finally, In the case of both hands, one hand is recognized and processed separately, such that its motion detection does not affect the other hand's motion detection, unless the two hands combine to form a single gesture, such as locking fingers together.

3.4 Limitation

At present, a number of sensors for the tracking of the hand in virtual reality devices are being developed. Consequently, the tracking range corresponds to the maximum viewing angle of the device. When shooting an arrow, one hand with the arrow faces forward, while the other hand pulling the string can extend beyond the back of the head. However, since hand tracking is employed in this implementation, it was necessary to implement an archery motion within the scope of the device's recognition of hand tracking.

4. Conclusion

In this paper, we made it possible to recognize archery gestures with the XR Hand provided by Unity. We adopted the archery movement as an action in this paper because it requires complicated hand movements and has historical significance. The package used is Unity's XR Hand, which is OpenXR-based and easy to handle. This implementation test was conducted on the Oculus Quest 2. Gesture detection took an average of 24.8 microseconds, and if accurate detection was desired, it was necessary not to add an intermediate hand motion or to maintain one hand motion for a certain period of time when implementing multiple gestures. In addition, there was a limit to the recognition range and accuracy because we used hand tracking that relied only on sensors in the HMD, rather than building a room for VR. In future work, we will introduce artificial intelligence (AI) for gesture detection and compare the performance between method with AI and without AI.

Acknowledgements

This study was supported by research fund from Chosun University, 2024.

References

[1] Apple, Apple Vision Pro, Apple Vision Pro Overview, available: https://www.apple.com/apple-vision-pro/

[2] The Khronos Group, OpenXR, available: https://www.khronos.org/openxr/

[3] Valve, OpenVR, Steamworks Documentation, available: https://partner.steamgames.com/doc/features/steamvr/openvr

[4] S.Li, Notification System for Unity OpenXR, dissertion, California Polytechnic State University, 2023

[5] M.Huzaifa et al., ILLIXR: An Open Testbed to Enable Extended Reality Systems Research, IEEE Micro, 42(4):97-106, 2024

[6] C.Runde, XR Standardization. A Global Overview, EuroXR 2023 Conference, 2023

[7] W3C, WebXR Device API, W3C Candidate Recommendation Draft, available: https://www.w3.org/TR/webxr/

[8] K.Jinkyu, OpenXR and WebXR in virtual augmented reality, Broadcast and Media, 26(1):12-18, 2021

[9] V.Juránek, Virtual reality toolkit for the Unity game engine, dissertation, Masaryk University, 2021

[10] American Society for Surgury of the Hand, Joints – finger joints, Handcare available: https://www.assh.org/handcare/safety/joints

[11] M.Tavakoli, R.Batista, and L.Sgrigna, The UC Softhand: Light Weight Adaptive Bionic Hand with a Compact Twisted String Actuation System, actuators, 5(1), 2015

[12] The Khronos Group, Conventions of hand joints, The OpenXR specification, available: https://registry.khronos.org/OpenXR/specs/1.0/html/xrspec.html#XR_EXT_hand_tracking

[13] Unity, Hand data model, Unity Manual, available: https://docs.unity3d.com/Packages/com.unity.xr.hands@1.1/manual/hand-data/xr-hand-data-model.html

[14] Unity, Finger shape, Unity Manual, available: https://docs.unity3d.com/Packages/com.unity.xr.hands@1.4/manual/gestures/finger-shapes.html

[15] Unity, Hand orientation, Unity Manual, available: https://docs.unity3d.com/Packages/com.unity.xr.hands@1.4/manual/gestures/hand-orientation.html

〈저자소개〉

조범준
- 2015-2020; Bachelor in Daegu University
- 2021-present; Integrated Ph.D. program, Department of Game Design and development, Sangmyung University
- Research interest; Game Design, Game Development
- https://orcid.org/0009-0006-3967-1372

김성기
- 2002-2009; Ph.D. in Computer Science and Engineering, SNU
- 2009-2014; Samsung Electronics
- 2017-2020; Assistant Professor, Keimyung University
- 2020-2024; Assistant Professor, Sangmyung University
- 2024-Present; Associate Professor, Chosun University
- Research Interest; Graphics, Mobile Device, Virtual/ Augmented Reality
- https://orcid.org/0000-0002-2664-3632