學士學位論文

사용자 간 협업을 위한 Eye 및 Hand Interaction 기반 XR Co-Working Space

전남대학교

공과대학
소프트웨어공학과

정　　재　　준

指導敎授　김　승　원

2024年 2月

사용자 간 협업을 위한 Eye 및 Hand Interaction 기반 XR Co-Working Space

전남대학교

공과대학
소프트웨어공학과

정 재 준

指導敎授(지도교수) 김 승 원 (인)

主任敎授(주임교수) 유 석 봉 (인)

2024年 2月

- 목 차 -

1. 서론

1.1 배경 및 문제

본 연구에서는 눈을 활용한 기술 연구의 증가와 AR/VR 기술의 출하량 감소와 활용성 부재에 대해 분석하였다.

가상(Virtual Reality: 이하 VR) 및 증강 현실(Augmented Reality: 이하 AR) 시스템에서 사용자는 가상의 물체를 선택 및 조작하는 등의 상호작용을 수행한다. 이러한 상호작용에서 가장 자연스럽고 직관적인 방법은 현실의 손 움직임을 가상의 손에 매핑시키는 Direct Hand Gesture이다. 그러나 이 인터페이스는 팔이 뻗어지는 범위로 가상 손의 작업 범위를 한정 짓는 단점이 있어 원거리 물체와 상호작용에 한계를 갖는다. 이러한 문제를 해결하기 위해, 최근 학계 및 산업계에서는 Apple 사의 HMD 눈 추적 기능과 같은 눈을 활용한 연구가 활발히 이루어지고 있다. 이는 신체적 활동을 최소화하면서 사용자의 관심사를 쉽게 공유할 수 있는 눈의 특성을 활용한 것이다.

한편, 2021년에 코로나19로 인해 집에 머무는 시간이 길어지면서 AR/VR 기기의 출하량이 매우 증가하였으나, 대부분 콘텐츠 소모에 초점이 맞춰 있는 시장의 특성으로 인해 2022년에는 감소하는 현상이 나타났다. 대부분의 AR/VR 서비스는 게임에 치우쳐져 있는데, 이는 많은 대체품이 존재하고, 쉽게 질리기 때문에 시장 자체에 대한 관심도가 떨어진 것으로 보인다.

이에 본 논문은 실감과 집중력이 떨어지는 문제를 가진 기존의 비대면 회의 시스템을 개선하기 위해 VR 기반의 원격 회의 시스템을 제안하고자 한다. 또한, 눈을 활용해 회의 시스템의 단점인 공존감의 부재를 극복하고 신체적 부담을 줄인 인터페이스를 도입하고자 한다. 여기에 현실 공간에 PC 화면과 위젯을 통해 나만의 업무 공간을 구성하고 필요시에 회의 때 공유할 수 있는 AR 업무 보조 시스템을 추가하고자 한다. 이 연구를 통해 AR/VR 기술의 활용성과 사용자 경험이 향상될 것으로 기대된다.

1.2 프로젝트 목표

본 연구에서는 사용자가 편리하게 원거리 오브젝트를 조작할 수 있는 AR/VR 상호작용 인터페이스 개발한다. 본 인터페이스는 눈의 편리함과 손의 직관성을 결합한 것으로, 눈이 일종의 마우스 포인터 역할을 하여 바라보는 물체에 가상 손이 이동하고, 이동한 손으로 원거리 물체를 잡아 조작할 수 있게 한다. 또한 사용자 실험을 통해 정량적, 정성적 평가를 통해 인터페이스 성능을 검증하고, 검증된 결과를 기반으로 추후 연구를 제안한다. 연구의 목표는 인터페이스 기반 1인용 작업 환경 시스템과 AR/VR 연계형 회의 시스템을 개발하는 것이다.

본 시스템을 통해 기존의 원거리 물체 조작 인터페이스의 성능 향상과 기존 회의 시스템의 몰입도 등의 단점 보완을 기대할 수 있다. 또한, 사용자가 보고 있는 위치를 표시함으로써 원활한 의사소통과 1인용 AR 개인 공간으로 공간적 제약 벗어남과 동시에 개인 정보 보호를 기대할 수 있다.

2. 관련 연구

2.1 Eye 및 Hand interaction

가상 및 증강 현실에서 원거리 물체를 조작하기 위해 다양한 연구가 진행되어왔다. 가장 대표적인 연구로 팔의 길이를 늘여 가상 손이 닿는 범위를 넓히는 Go-Go 인터페이스가 제시되었다 [1]. 하지만 늘어난 팔의 길이만큼 실제 손을 조금만 움직여도 가상 손이 민감하게 반응하여 정교한 원거리 물체 조작이 어렵다는 단점을 가진다 [7].

다른 연구자들은 시점 상호작용을 사용한 물체 조작 방법을 시도하였다. Liu 등[3]은 선택된 물체를 시점 상호작용을 통해 이동시키는 방법론을 제시하였다. 그러나 시점 상호작용을 통한 이동 및 조작은 6-자유도 (degree of freedom) 움직임을 지원하지 않았으며 현실 세계에서의 물체 조작법과 상이하여 실감이 떨어진다.

최근 손과 시점 상호작용을 결합하여 원거리 물체를 조작하는 인터페이스들이 연구되었다 [2, 4, 5]. Pfeuffer 등 [2]은 시점이 물체에 닿은 상태(사용자가 물체를 보고 있는 상태)에서 손 Pinch 동작을 사용하여 물체를 선택하고, 양손 Pinch로 물체를 선택한 상태에서 양손의 상대 위치로 물체의 크기 및 위치, 방향을 조작하는 방법을 제시하였다. 하지만, 이 방법은 물체 조작을 위해 손을 항상 Pinch 동작으로 유지해야 했으며 이는 자연스러운 손 상호작용을 방해하는 요소로 작용하였다. 류건희 등[5]은 시점을 통해 영역을 선택하고 선택하려는 물체의 너비만큼 손을 벌려 그 영역 안의 물체 중 하나를 선택하였다. 또한, 선택된 물체의 조작을 위해 Direct Hand Gesture 상호작용 방법을 사용하였다.

이렇게 다양한 원거리 물체 상호작용 방법들이 시점 및 가상 손을 바탕으로 제시되었지만, 여전히 문제점들을 가지고 있다. 본 논문에서는 이러한 문제점을 해결하고, 빠르며 사용 범위에 한계가 없는 시점 인터페이스의 장점과 자연스럽고 직관적인 손 상호작용의 장점을 결합한 새로운 형태의 원거리 물체 상호작용 기술을 제시한다.

2.2 기존 서비스 분석

본 연구에서는 기존 회의 시스템과의 비교를 통해 제안하는 시스템의 차별성을 분석한다. 기존 회의 시스템으로는 2D 영상 기반의 Zoom, VR 환경에서 아바타를 제공하는 Horizon Workrooms, 그리고 AR 환경에서 가상 오브젝트를 공유하는 Spatial 등이 있다. 이들 시스템은 각각의 방식으로 사용자들에게 회의 환경을 제공하지만, 몰입도와 원거리 물체 상호작용에 있어서 한계를 가진다.

Zoom과 같은 기존 화상 회의 시스템은 2D 영상 통신 방식으로 사용자의 환경 공유 및 사용자의 작업 몰입도와 집중력에 한계를 가지고 있다. 이러한 문제점을 해결하기 위해 Horizon Workrooms나 Spatial처럼 VR/AR 회의 솔루션이 소개되었다. 이는 앞서 언급한 것처럼 상대방의 행동과 동기화된 아바타를 보며 회의하여 몰입감은 높였지만, 상대방이 어디를 보고 있는지 알 수 없으며, 원거리 물체 상호작용(배치, 조작 등)이 어렵다는 단점이 존재한다. 본 연구에서는 이러한 회의 시스템의 공존감 및 집중력 문제를 해결하기 위해 XR(Extended Reality: 혼합 현실) 환경을 선택하였고, 기존 VR/AR 회의의 단점을 극복하기 위한 새로운 눈+손 인터페이스를 제안한다. 또한 1인용 작업 환경 시스템을 개발하여 개인 업무를 AR 환경에서 작업하고, 작업물을 VR 회의 시스템에 가져올 수 있도록 하는 회의 시스템 “Emptied”를 제안한다.

다시 말해, 제안하는 Emptied는 XR 회의 시스템으로 상대방의 행동과 동기화된 아바타를 제공하면서 원활한 원거리 물체 조작을 위해 눈+손 인터페이스를 제공한다. 이는 VR 회의에서 상대방이 보고 있는 곳을 확인하는 데에도 사용된다. 또한, 개인 업무를 위한 1인용 AR 시스템이 존재하면 작업물을 VR 회의 시스템에 가져올 수 있다.

3. 시스템 설계 및 구현

3.1 사용자 요구사항

요구사항은 주로 기능적 요구사항을 중심으로 작성하였다. 비기능적 요구사항의 경우, 우선 1~4인 수준에서 원활하게 작동하는 수준으로 설정하였다.

요구사항 정의서				
분류	**번호**	**개요**	**설명**	**유형**
Interface	IF-1	제자리에서 Hand Tracking으로 원거리 가상 물체를 선택 및 조작	사용자는 몸을 움직이지 않고도 멀리 있는 가상 물체(메모장, 스케치 등)를 선택하고 조작할 수 있다.	**필수**
	IF-2	제자리에서 가상 물체를 원거리로 이동	사용자는 몸을 움직이지 않고도 가까운 가상 물체를 원하는 곳(멀리 있는 곳)으로 이동시킬 수 있다.	**필수**
	IF-3	신체적 움직임을 줄여주는 인터페이스	사용자는 기본 가상 손 인터페이스를 사용할 때보다 더 적은 움직임으로 같은 목적을 달성할 수 있다.	**필수**
AR workspace	AR-1	PC 화면 불러오기	사용자는 본 시스템을 자신의 PC와 연결함으로써 그 화면을 볼 수 있다.	**필수**
	AR-2	카메라를 통한 현실 입력	사용자는 기기를 착용한 상태에서도 영상을 통해 현실을 볼 수 있다.	**필수**
	AR-3	위젯(미니 애플리케이션) 구현	사용자의 필요에 따라 메모장, 캘린더, 스톱워치, 사진액자 등을 배치할 수 있다.	**필수**
	AR-4	음성으로 메모 작성	사용자는 음성 입력을 통해 메모를 작성할 수 있다.	**필수**
	AR-5	객체 상태 저장	사용자가 가상 오브젝트를 배치한 후, 본 애플리케이션을 종료하고 재접속하여도 가상 오브젝트는 해당 위치에 보존된다.	**필수**
	AR-6	현실을 고려한 가상 물체 배치	시스템은 현실 구조물(책상 등) 위치 및 모양을 인식하여 사용자가 그 위에 가상 물체를 배치하도록 도울 수 있다.	**선택**
	AR-7	다양한 방 설정 저장	사용자는 집, 사무실 등 자신의 업무 환경에 따른 여러 개의 물체 배치를 저장할 수 있다.	**선택**
VR meeting	VR-1	회의 방 생성	사용자는 회의를 위한 온라인 VR 회의 방을 생성하여 사용자를 초대하고, 동시에 상호작용할 수 있다.	**필수**
	VR-2	상대방이 어디를 보고 있는지 확인	모든 사용자는 각 사용자가 어디가 보고 있는지 gaze pointer를 통해 확인할 수 있다.	**필수**
	VR-3	AR workspace와의 연동 (Object)	사용자는 자신의 개인 AR workspace에서 만들어뒀던 물체(내 메모, 캘린더 등)를 VR 회의 환경으로 불러와 함께 공유할 수 있다.	**필수**
	VR-4	PC 화면 다수와 공유	WebRTC를 활용하여 사용자는 공유된 자신의 PC 화면을 자신뿐 아니라 다른 사람과도 공유할 수 있다.	**필수**
	VR-5	보이스톡 기능	사용자는 회의 중 1명 또는 불특정 다수에게 음성으로 자신의 의견을 전달할 수 있다.	**필수**
	VR-6	회의 참여자 아바타 보기	사용자는 서로를 알아볼 수 있는 가상 아바타로 회의 참여가 가능하다.	**선택**
	VR-7	회의 초대 기능 구현	회의 호스트는 다른 유저에게 초대 메시지를 보낼 수 있으며, AR workspace에 있는 유저가 VR 환경으로 들어갈 수 있게 우측 상단에 초대 메시지를 표시한다.	**선택**
AR & VR 모두 (XR)	XR-1	선택된 가상 물체의 자유로운 조작 및 배치	사용자는 선택된 가상 물체를 자유롭게 이동, 회전, 확대/축소할 수 있다.	**필수**
	XR-2	3D 스케치 추가	사용자는 AR/VR 환경에서 활용할 수 있는 입체적인 스케치를 추가할 수 있다.	**필수**
	XR-3	가상 물체 고정	사용자는 시스템 사용 중 의도치 않은 물체 상호작용 및 손실을 방지하기 위해 일시적으로 물체를 고정할 수 있다.	**필수**
	XR-4	가상 물체의 속성 수정	사용자는 물체의 크기와 색상 등 허용되는 가상 물체의 속성을 생성된 후에라도 수정할 수 있다.	**선택**
	XR-5	키보드로 메모 작성	사용자는 블루투스 키보드 입력 (or 가상 키보드)을 받아 메모를 작성할 수 있다.	**선택**
	XR-6	AR/VR 전환	사용자는 현재 모드에서 반대 모드로 뷰를 바꿀 수 있다.	**선택**

표 1. 요구사항 정의서

3.2 시스템 설계

3.2.1 종합 아키텍처

본 프로젝트의 아키텍처는 2가지 갈래로 나누어 생각할 수 있다. 하나는 시스템의 구성요소(Web, AR, VR) 측면에서 생각하는 것이며, 다른 하나는 사용자가 시스템을 사용하는 흐름 측면에서 생각하는 것이다. 2가지 아키텍처를 생각함으로써 다각도에서 시스템을 계획할 수 있었다.
아래는 시스템의 구성요소 측면에서의 아키텍처를 나타낸다.

그림 1. 시스템 구성 요소 기준 아키텍처

아래는 시스템을 사용하는 흐름 측면에서의 아키텍처를 나타낸다. 사용자가 애플리케이션을 사용할 때 뒤에서 WebRTC, Photon Fusion 서버 등이 유기적으로 돌아가 서비스를 제공함을 알 수 있다.

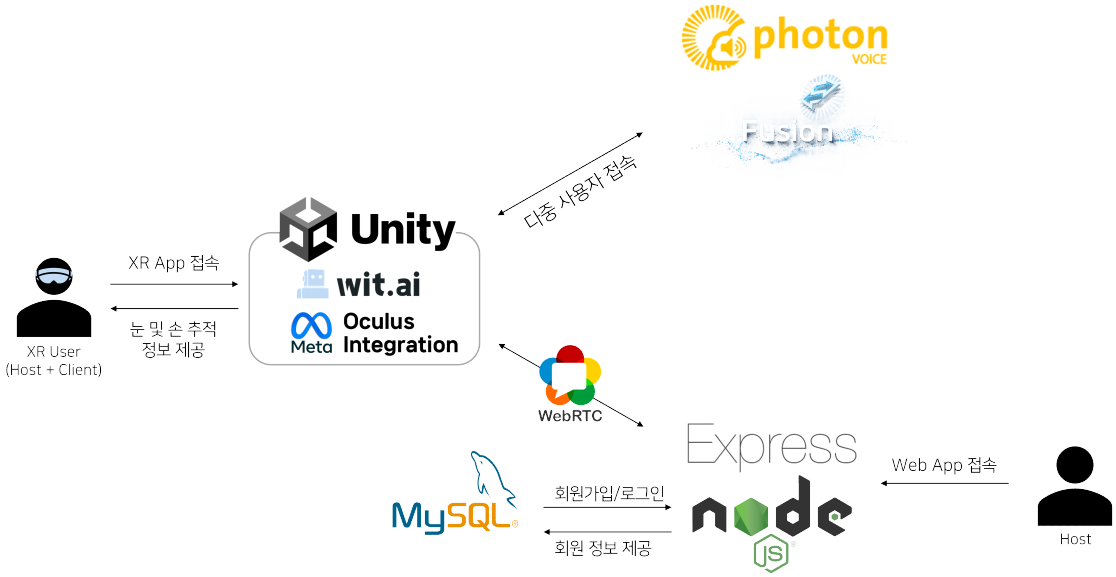

그림 2. 시스템 사용 흐름 기준 아키텍처

3.2.2 UML 다이어그램

유스케이스, 실행 흐름도, 클래스 다이어그램 등을 활용하여 시스템 구현 흐름을 이해하고 개발 방향을 수립하였다.

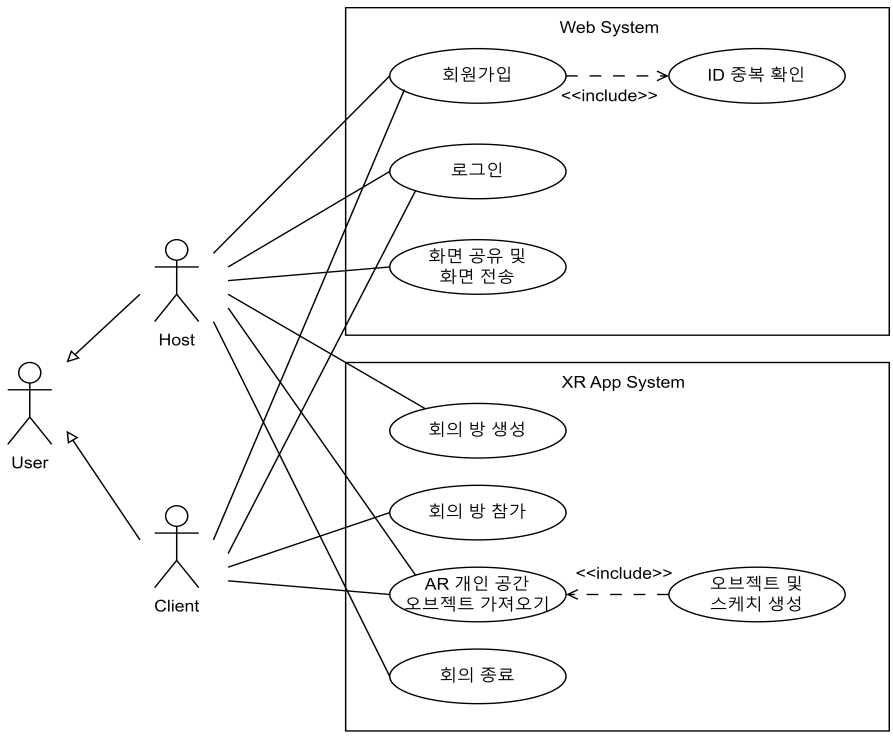

그림 3. 유스케이스 다이어그램

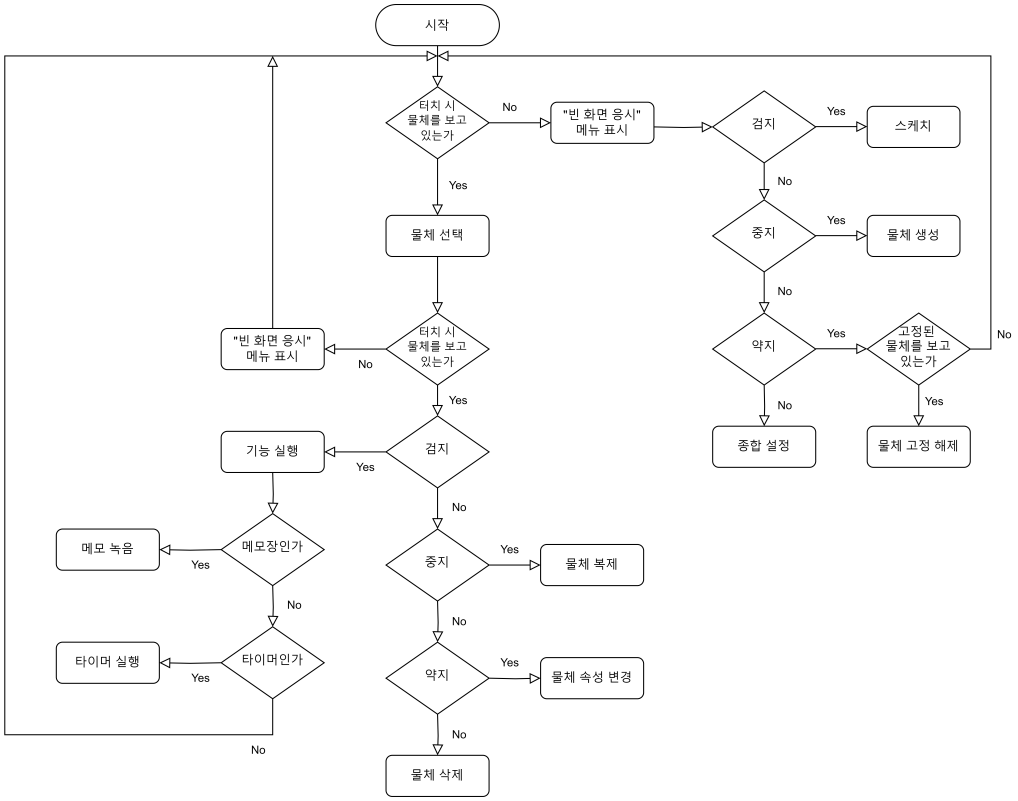

그림 4. AR 환경에서 기능 실행을 위한 흐름도

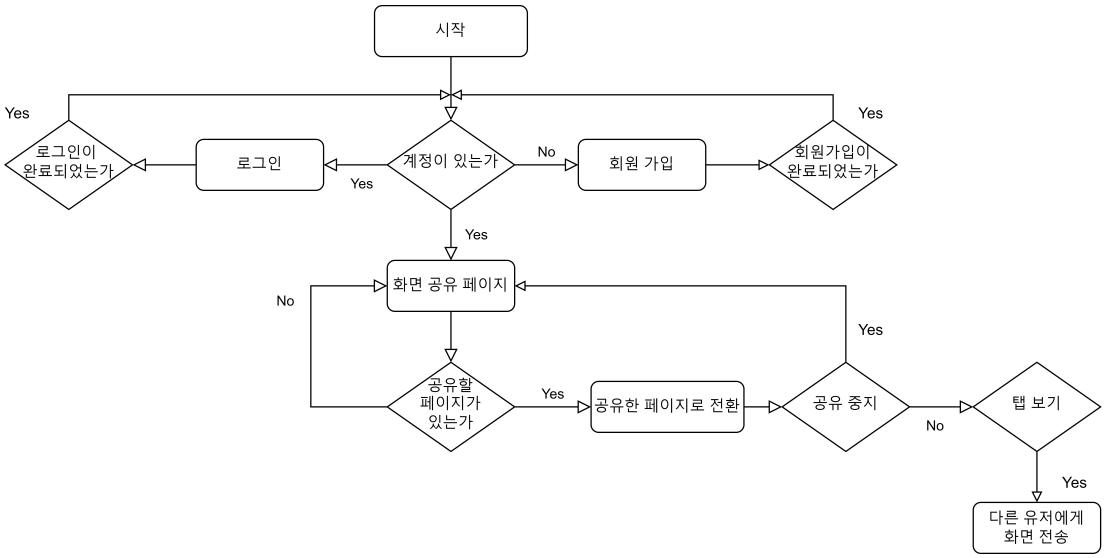

그림 5. 웹 앱에서 시작부터 화면 공유까지의 흐름도

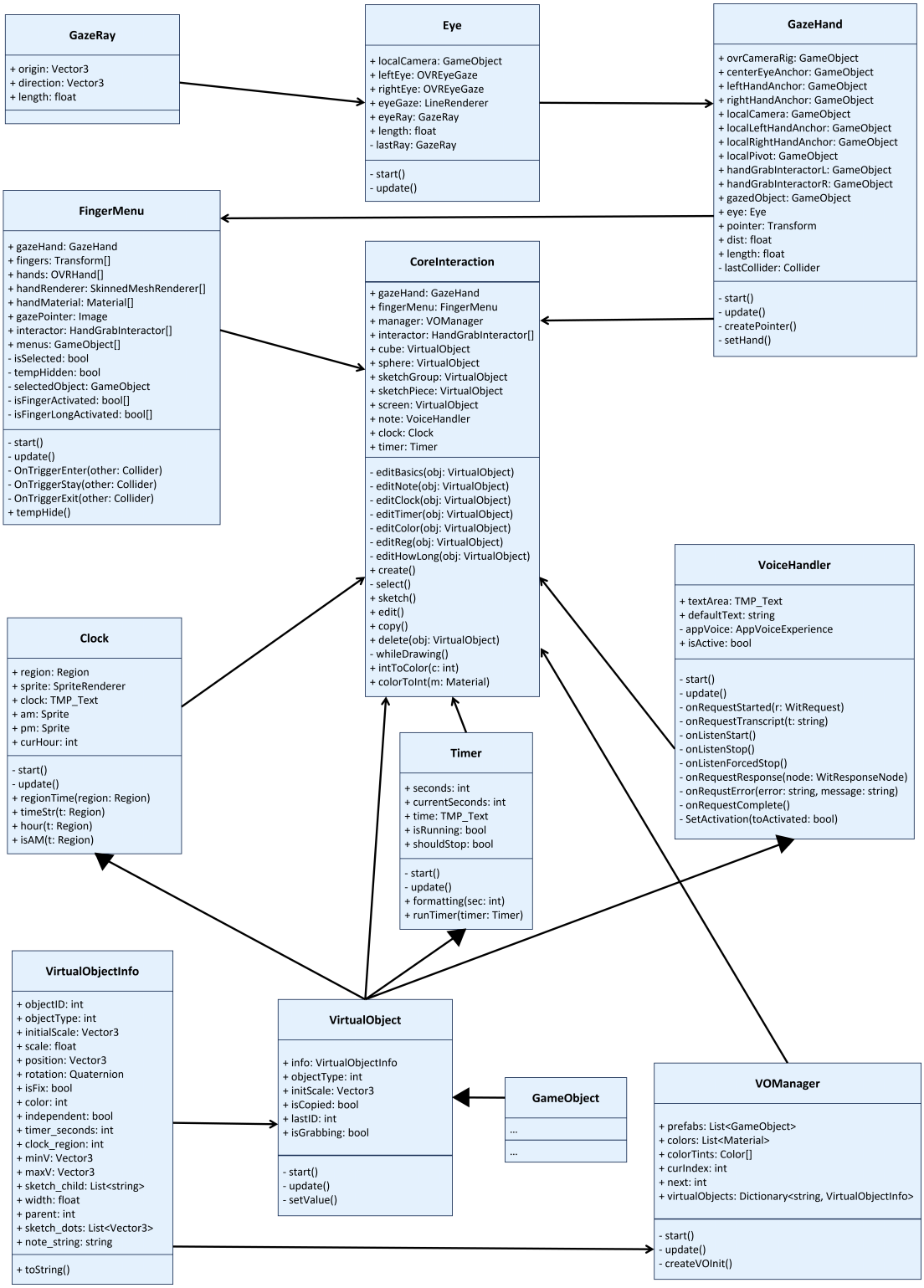

그림 6. AR 시스템을 설명하는 클래스 다이어그램

3.3 구현

3.3.1 구현 환경(사용한 도구, OS, 언어)

본 시스템의 웹 기반 구성요소는 HTML, CSS, JavaScript을 사용하였으며, 벡엔드 환경으로는 Node.js 16.16.0와 Express 4.18.2를 사용하였다. EJS 템플릿 엔진을 통해 서버 사이드 렌더링이 가능하게 되었으며, WebRTC를 통해 브라우저 간 실시간 통신이 구현되었다. 데이터 관리는 MySQL 8.0.33을 통해 수행되었으며, OpenSSL을 사용하여 데이터 전송의 보안을 강화하였다.

AR/VR 인터페이스 개발은 Unity 2021.3.21.f1을 사용하였고, Oculus Integration SDK 50를 기반으로 Meta Quest Pro 기기에서 제공하는 gaze 및 손 추적 결과를 사용하여 구현하였다. 한편, AR 인터페이스에서 음성인식 기능은 Wit.ai를 통해 구현하였다. VR 회의 시스템 구축을 위해 Unity Render Streaming 3.1.0-exp6, Photon Fusion 1.1.6, Photon Voice 2.52 등의 라이브러리와 툴을 채택하였다.

또한 VR 회의 시스템에 활용할 3D 환경 개발에 회의실 모델[9]과 3D 아바타 모델[10]을 사용하였다.

3.3.2 제안 시스템 아이덴티티

(a) (b) (c)

그림 7. (a) XRill 팀 로고 (b) emptied 로고 (c) emptied 앱 아이콘

제안하는 서비스의 통합된 이미지를 제시하기 위해 팀명과 로고를 디자인했다. 우선 'XRill'이라는 팀명은 확장 현실(eXtended Reality)을 의미하는 XR과 전율을 의미하는 영어 단어 thrill의 합성어로 다가올 확장현실을 기반으로 한 미래에 사람들에게 전율을 선사한다는 의미로 결정했다. 'XRill'의 로고는 전율이라는 의미를 잘 전달할 수 있도록 자동차 스키드마크가 연상되게 디자인하였다. 다음으로 'emptied'라는 이름은 사람들이 책상을 비우고 가상 오브젝트를 활용해 업무를 하며 효율을 끌어올리기를 바라면서 결정했다. 로고의 빨강, 파랑, 노랑은 emptied만의 다채로운 가상 오브젝트들을 추상화한 것이다.

3.3.3 사용자 인터페이스(UI) 구현

손가락에 부착되는 메뉴 및 위젯 UI는 다음과 같다.

사용자의 온전한 업무 몰입을 돕기 위해 아이콘과 오브젝트들을 최대한 단순하고 직관적으로 디자인했다.

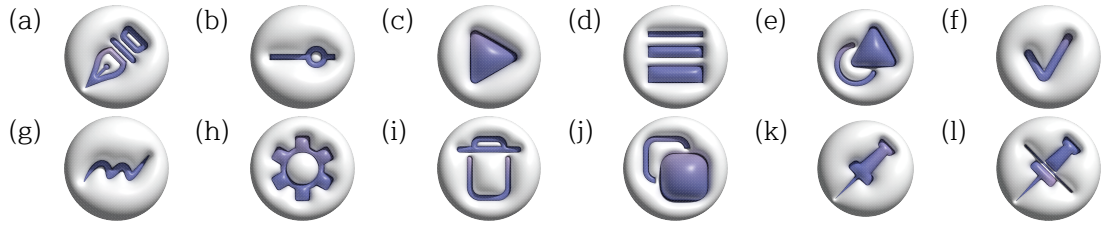

그림 8. (a) 펜 스타일 (b) 색상 조정 (c) 재생 (d) 굵기 조정 (e) 도형 (f) 확인 (g) 스케치 (h) 설정 (i) 삭제 (j) 복사 (k) 고정 (l) 고정 해제

위 이미지는 사용자가 얼굴을 마주 보고 손바닥을 펼쳤을 때 손가락 위에 나타나는 버튼이다. 검지, 중지, 약지, 새끼손가락에 각 버튼이 나타나고 엄지손가락과 맞닿으면 그 아이콘에 해당하는 기능이 실행되는 방식이다. 이와 같은 방식을 통해 사용자가 자주 쓰는 기능을 실행하는 방법을 쉽게 기억하여 업무의 몰입을 도울 수 있다.

그림 9. (a) 캘린더 (b) 시계 (c) PC 화면 스트림 (d) 음성 메모

위젯 아이콘은 현대인이 자주 접하는 스마트폰의 앱 아이콘을 모방하여 디자인하였다. AR 환경에서 위 아이콘들을 손으로 집어와 가상 공간에 배치할 수 있다.

Web-app UI는 다음과 같다.

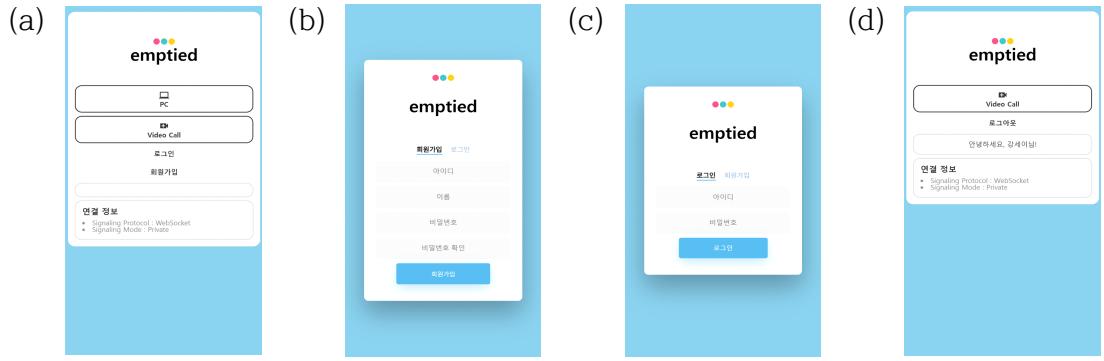

그림 10. (a) 첫 화면 (b) 회원가입 화면 (c) 로그인 화면 (d) 로그인 성공 화면

Web-app은 유저가 자신의 PC 화면을 가상 공간에 송출할 수 있는 기능을 제공한다. 로그인하여 사용자 인증 후 같은 네트워크에 접속해 있는 HMD 디바이스로 현재 PC 화면을 WebRTC를 통해 전송한다.

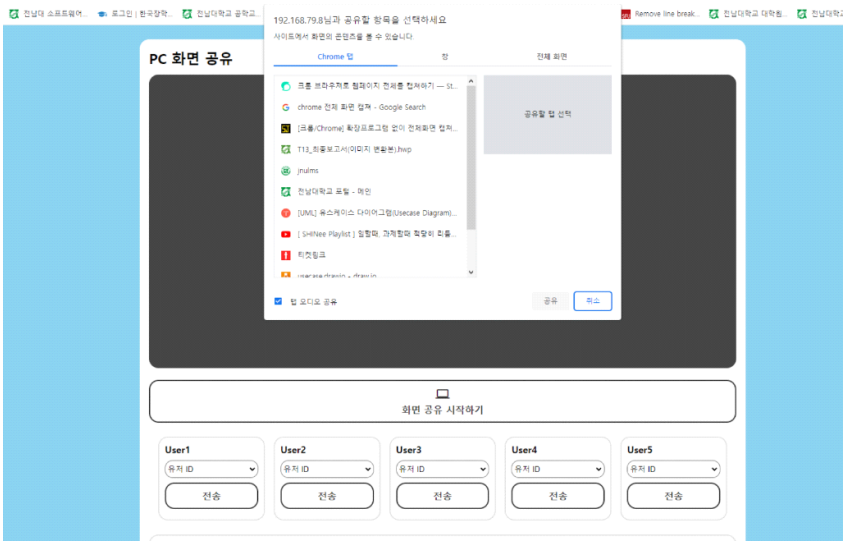

그림 11. PC 화면 전송 UI

Web-app에서 PC 화면 공유 메뉴로 들어가 유저 ID를 선택한다. 여기서 유저 ID는 HMD 상에 로그인한 유저를 의미한다. 화면 공유 시작하기 버튼을 클릭하면 Chrome에서 어떤 화면을 전송할지 묻는 대화상자가 나타난다. 거기서 전체 화면을 전송할지, 특정 창을 전송할지 선택한 후 확인을 클릭하면 영상이 전송된다.

AR 환경에서 사용자가 맞이하는 UI에는 현재 개설된 방의 종류와 자신이 속한 팀의 팀원 목록을 볼 수 있도록 하였다. 각 목록은 상하 버튼을 통해 탐

색할 수 있게 하였다. 또한 새로운 회의 방을 생성하기 위한 New Meeting 버튼을 하단에 추가하였다.

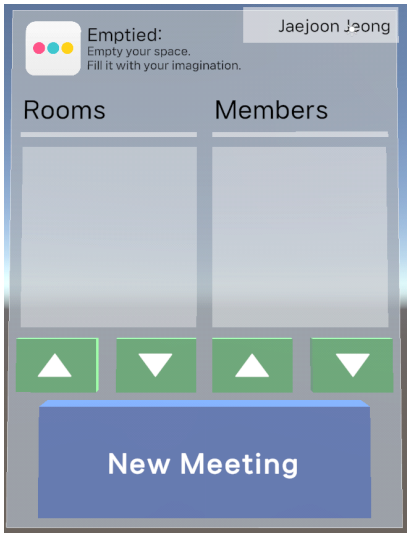

그림 12. AR Scene에서의 메인 메뉴 UI

3.3.4 상호작용 인터페이스 구현

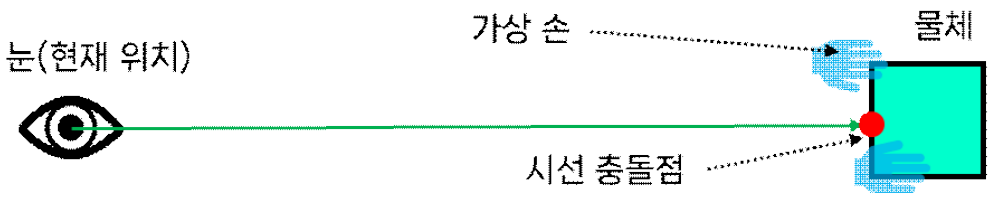

그림 13. 상호작용 인터페이스 평면도

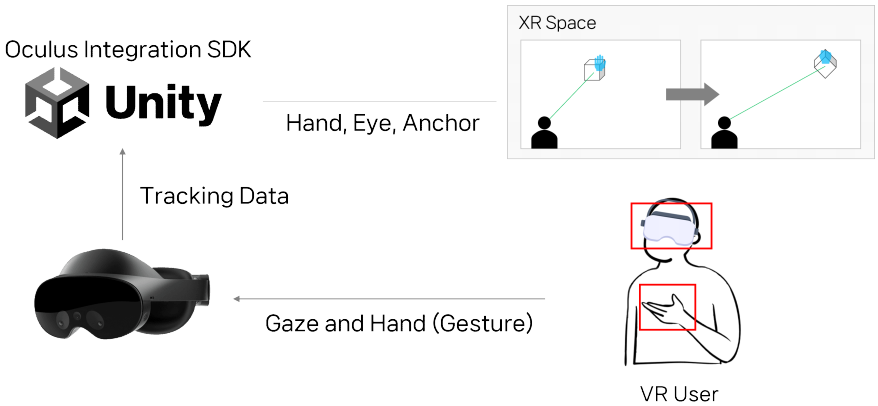

그림 14. 인터페이스 시스템 구조도

본 인터페이스는 Oculus Integration의 Hand Grab Interactor와 OVREyeGaze 기반으로 구현하였다. 이 인터페이스는 사람의 시선을 나타내는 직선과 포인터, 그리고 눈앞에 있는 현실의 손과 물건을 잡을 수 있는 원거리 상호작용 손의 짝으로 구성된다.
인터페이스를 활용한 물체 상호작용 방법과 순서는 다음과 같다. 우선 사용자의 시선을 출발점으로 하는 광선을 발사하여 최초 충돌한 물체와의 거리를 측정한다. 이것을 시선 거리(Gaze Ray Depth)로 정의한다. 이어 사용자의 팔 길이를 고려하여 충돌 지점으로부터 일정 거리 앞에 기준점을 설정한다. 해당 기준점은 사람의 몸이 위치할 자리로 가정한다. 이렇게 하는 이유는 물체에 몸이 위치하면 둘이 겹치고 손이 물체보다 뒤에 있어 물체를 제대로 잡을 수 없게 되기 때문이다. 이렇게 생성된 기준점에 사용자가 있다고 가정하고 이곳에 원거리 손을 배치하면 물체 바로 앞에 손이 위치하게 되므로 물체를 바로 잡을 수 있게 된다. 만약 사용자 전방에 물체가 없어 시선 거리를 측정할 수 없는 경우, 마지막 시선 거리가 보존된다.

3.3.5 AR 워크스페이스 구현

AR 워크스페이스 구현에서 주로 고려한 사항은 2가지이다. 하나는 시스템에 물체 정보를 저장하는 방법, 다른 하나는 물체와 상호작용하는 방법이다.
첫 번째로, 시스템에 물체 정보를 저장하는 방식을 고려하였다. 본 시스템은 로컬에도 정보를 저장하지만 네트워크를 통해서 현재 정보가 전달되어야 하므로 두 가지 케이스를 모두 고려하기 위해 텍스트 형태로 전송할 수 있는 JSON을 활용하기로 하였다. 이때 오브젝트 하나를 식별할 때 여러 개의 JSON을 만드는 대신, 공통 필드와 선택 필드로 구분된 하나의 JSON을 정의하여 모든 오브젝트에 대응되도록 하였다. 용량을 최적화하기 위해 중복 정보는 최대한 피하고, 스케치 등 자식이 딸린 오브젝트의 경우 자식의 고유 ID만 저장하였다. 그리고 구현을 단순화하기 위해 기본으로 제공하는 자료형을 최대한 활용하였다. 표 2는 본 시스템에서 가상 오브젝트를 정의하는 데 사용한 필드들을 나타낸다.

필드	타입	설명	공통 / 개별
objectID	int	물체 고유 번호 (오브젝트 저장 시 활용)	공통
objectType	int	물체의 유형 (종류에 따라 0~8)	공통
initialScale	Vector3	각 물체별 최초 크기 저장	공통
scale	float	현재 크기인 실수배값 저장 (실제 크기는 scale*initialScale)	공통
position	Vector3	물체의 위치	공통
rotation	Quaternion	물체의 회전	공통
color	int	물체 색상 (종류에 따라 0~7)	공통
independent	bool	서브스케치 이외에는 true (서브스케치는 스케치 그룹에 딸려 생성되어 종속되므로 false)	공통
isFix	bool	물체 고정 여부	공통
timerSeconds	int	타이머 설정 시간(초)	타이머
clockRegion	int	시계 설정 구역(0~3)	세계시간
minV	Vector3	스케치 바운딩 박스의 최소 지점	스케치
maxV	Vector3	스케치 바운딩 박스의 최대 지점	스케치
sketchChildID	string[]	서브스케치의 고유 번호 배열 (단, int를 string으로 변환하여 저장)	스케치
width	float	서브스케치 두께	서브스케치
parent	int	서브스케치의 부모 ID	서브스케치
sketchDots	Vector3[]	서브스케치의 점들의 모음	서브스케치
noteString	string	메모장에 기록된 텍스트	메모장

표 2. Emptied의 JSON 기반 물체 속성

이렇게 정의된 물체 속성은 JSON 형태로 추출되어 시스템 내 저장소에 저장되거나 네트워크를 통해 전송된다.

{"objectID":199,"objectType":0,"initialScale":{"x":0.6994391083717346,"y":0.6994391083717346,"z":0.6994391083717346},"scale":1.0,"position":{"x
{"objectID":201,"objectType":3,"initialScale":{"x":0.800000011920929,"y":0.3999999761581421,"z":0.009999999776482582},"scale":1.0,"position":{"
{"objectID":202,"objectType":3,"initialScale":{"x":0.800000011920929,"y":0.3999999761581421,"z":0.009999999776482582},"scale":1.0,"position":{"
{"objectID":203,"objectType":6,"initialScale":{"x":0.5983312726020813,"y":0.5983312129974365,"z":0.014958281069993973},"scale":1.0,"position":
{"objectID":204,"objectType":6,"initialScale":{"x":0.5983312726020813,"y":0.5983312129974365,"z":0.014958281069993973},"scale":1.0,"position":
{"objectID":222,"objectType":2,"initialScale":{"x":1.139133095741272,"y":0.5695664882659912,"z":0.014239163137972355},"scale":1.0,"position":{"x
{"objectID":245,"objectType":3,"initialScale":{"x":0.800000011920929,"y":0.3999999761581421,"z":0.009999999776482582},"scale":1.0,"position":{"
{"objectID":268,"objectType":6,"initialScale":{"x":2.2588558197021486,"y":2.2588555812835695,"z":0.05647139251232147},"scale":1.0,"position":{"x"
{"objectID":286,"objectType":4,"initialScale":{"x":4.2702131271362309,"y":4.2702131271362309,"z":4.2702131271362309},"scale":1.0,"position":{"x"

```
void Start()
{

    keyList = JsonUtility.FromJson<Keys>(PlayerPrefs.GetString("Keys"));
    if (keyList == null)
        keyList = new Keys();
    virtualObjects = new Dictionary<string, VirtualObjectInfo>();
    curIndex = keyList.next;

    foreach(string key in keyList.keyList)
    {
        virtualObjects.Add(key, JsonUtility.FromJson<VirtualObjectInfo>(PlayerPrefs.GetString(key)));
    }

    // 불러와진 VO를 생성
    createVOInit();
}
```

그림 15. 위: 오브젝트의 속성이 기록된 JSON의 일부, 아래: JSON으로부터 가상 물체를 복원하는 C# 코드의 일부

한편, 현실감을 높이기 위해 가상 오브젝트를 실제 공간에 증강하기 위한 좌표 원점 Spatial Anchor를 추가 적용하는 테스트를 진행했다. Spatial Anchor 정보를 Playerprefs (Android XML 내장)에 저장하여 재부팅 해도 가상 오브젝트는 처음 배치한 위치에 그대로 표시되는 것을 알 수 있다.

그림 16. Spatial Anchor 적용 전후 사진

두 번째로, 가상 물체를 생성하고 배치하거나 생성된 물체와 상호작용하는 방법을 고려하였다. 사용자의 현실감을 높이고자 본 시스템에서는 컨트롤러 대신 손으로 모든 작업을 할 수 있도록 하였다.
먼저 각 기능을 손가락만으로 탐색하여 실행할 수 있도록 구현하였다. 이를

위해 각 손가락에 구형 Collider(충돌 감지 오브젝트)를 장착하였다. 이렇게 하면 충돌 이벤트를 기반으로 사용자의 제스처를 파악할 수 있다. 예를 들어 시스템 사용 중 엄지손가락 Collider와 다른 손가락 Collider가 맞닿으면 상황에 따라 메뉴가 실행된다. 손가락을 맞닿는다는 의미는, OK 제스처를 취하면 엄지와 검지의 끝이 서로 만나는 것을 의미한다. 같은 원리로 엄지손가락과 다른 손가락을 맞닿는 것으로 각 제스처를 구분할 수 있다. 한편, 손가락 추적 정확도 문제로 의도치 않은 연속 선택을 방지하기 위해 한 번 메뉴가 실행되면 0.5초간 선택되지 않도록 하였다.

기본적인 메뉴 사용 흐름은 다음과 같다. 예를 들어, '물체 복제' 기능을 사용하고 싶다면, 원하는 물체를 응시한 상태로 아무 손가락이나 맞닿으면 물체 주변에 녹색 테두리가 표시된다. 이 상태에서 물체를 계속 응시하며 엄지와 중지를 서로 맞닿으면 '복제' 기능이 실행되며 같은 특성을 갖는 새로운 물체가 생성된다. 전반적인 시스템 사용 방법은 아래 그림을 참고한다.

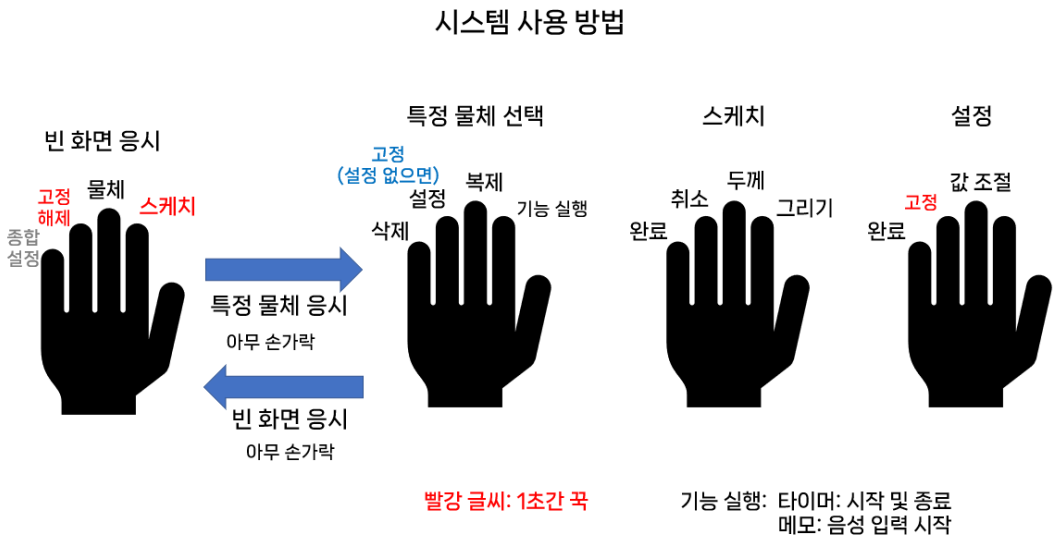

그림 17. 손가락을 활용한 Emptied 시스템 메뉴 사용 방법

생성되는 물체는 대부분 미리 정의되어 있다. 그러나 스케치는 사용자가 그리고자 하는 모양으로 그려져야 하는 문제가 있다. 즉, 일반 오브젝트와 달리 형태가 가변적이며, 특히 효과적으로 사용하려면 여러 스케치를 그룹화해야 하는 문제가 있어 메뉴 실행 시에도 특별한 오브젝트로 취급하였다. 그림 5에 표현된 방법대로 엄지와 검지를 맞닿아 스케치 기능을 실행하면 메뉴가 바뀌며 두께를 조절해가며 스케치를 그릴 수 있게 된다. 한붓그리기 방식 스케치가 아닌 여러 획으로 나누어 자유롭게 스케치할 수 있고, 완성한 후에는 그룹화되어 한 번에 잡고 움직일 수 있도록 하였다.

그림 18. 스케치 시연 및 두께 설정

스케치 이외에 구현된 오브젝트로는 단순 도형(큐브, 구), 메모장, 시계, 타이머, PC 스크린이 있다. 여기서 메모장, 시계, 타이머 등 UI 기반 오브젝트는 사용자에게 유용한 기능을 제공하는 위젯으로 분류된다.
메모장과 타이머의 경우 동적 위젯으로서 메모 작성이나 타이머 시작 기능을 실행할 수 있다. 기능 실행을 위해서는 물체를 선택한 상태로 엄지와 검지를 맞닿는다.
메모장은 Wit.ai API를 통해 Speech to Text 음성 메모를 지원하도록 했다. 사용자가 녹음 기능을 실행하면 녹음 모드로 전환되며 사용자의 음성이 Wit.ai 서버로 전송된다. STT 처리 후 텍스트 변환 결과가 실시간으로 메모장에 입력된다. 말을 멈추면 자동으로 변환을 멈춘다.
시계는 세계시간 정보를 제공하는 위젯으로 서울, 런던, 워싱턴, 시드니 등 4개국만 적용하였다. 시간대는 설정 메뉴로 진입하여 수정할 수 있다.
타이머는 미리 설정된 분과 초에 해당하는 시간을 카운트한다. 기능을 실행하면 0초가 되면 자동으로 멈추며, 0초가 되기 전에 터치하면 0초로 초기화된다. 시간 조절은 설정 메뉴로 진입하여 수정할 수 있다.
이러한 모든 오브젝트는 위에 설명된 상호작용 인터페이스를 통해 옮기거나 확대/축소/회전할 수 있다. 속성을 변경하려면 물체를 선택한 상태로 엄지와 약지를 맞닿아 설정 메뉴로 진입하면 된다. 단순 도형과 메모장의 경우 색상을 변경할 수 있으며, 모든 위젯은 가상 공간에 고정할 수 있다.

3.3.6 VR 워크스페이스 구현

프로젝트 목표에 맞게 다중 접속이 가능하도록 Photon Fusion 게임 서버와 연결하였다. VR 회의실의 방 이름은 호스트의 이름으로 결정되도록 하였다.

그림 19. Photon Fusion 관리 콘솔

현재 개설된 방 목록은 ARLobby.cs 스크립트에서 Fusion API를 통해 얻어오며, 방 목록이 UI에 표시되도록 하였다. 메뉴 UI에서 원하는 방을 응시한 상태로 엄지와 검지를 맞닿으면 해당 방으로 접속할 수 있다. 유저(팀원) 목록은 mySQL에 저장된 정보를 JSON 형태로 불러온 후, 지금 로그인한 유저를 제외한 나머지 유저 전체를 해당 UI에 표시하는 식으로 구현되었다.

그림 20. Photon과 MySQL을 이용한 AR 대기실 메뉴

회의실 내에서는 사용자의 현재 헤드셋 위치 정보와 시선 정보를 전송하여 사용자별로 서로 바라보고 있다는 느낌과 자연스러운 움직임을 나타내도록 하였다. 구현할 때는 원격 함수 호출(Remote Procedure Call, RPC) 기능을 활용하고, 각 사용자의 로컬 좌표 정보를 보내 서버에서 움직여주는 방식으로 구현하였다.

한편, VR 회의실 내에서 오브젝트를 생성하거나 공유하는 기능을 구현하는 부분이 가장 어려웠다. 사용자 혼자만 신경 쓰면 되는 AR 모드와 달리 VR 모드는 서버의 상태와 모든 사용자의 환경을 모두 고려해야 하기 때문이다. 특히 대부분의 네트워킹 API를 포함하여 Photon Fusion은 한 오브젝트에 동시 접근하는 것을 막고 있으며, 원칙적으로 회의 호스트가 모든 오브젝트의 권한을 갖는다. 다시 말해 모든 회의 접속자 중 회의 호스트만 오브젝트의 상태를 바꿀 수 있다. 이것이 한 오브젝트에 모두가 접근할 수 있어야 하는 본 시스템 구현을 어렵게 하는 원인이었다. 이를 해결하기 위한 기본적인 실행 흐름은 다음과 같다.

(1) 호스트나 참여자와 관계없이 사용자가 오브젝트에 접근하는 순간 로컬에서는 회의 호스트를 찾는다.

(2) 사용자는 로컬에서 자신의 의도대로 오브젝트를 변형하려고 하는데, 이때 해당 변형 정보를 RPC를 활용하여 실시간으로 회의 호스트에게 전송한다.

(3) 회의 호스트의 시스템은 해당 정보를 받으면 자신의 권한으로 해당 오브젝트를 그 모양으로 수정한다.

이렇게 하면 호스트가 자신의 권한으로 오브젝트에 변형을 준 거지만, 회의 화면에서 보기에는 상대방이 그렇게 한 것처럼 표시된다. 이를 정리하면 아래 그림과 같다.

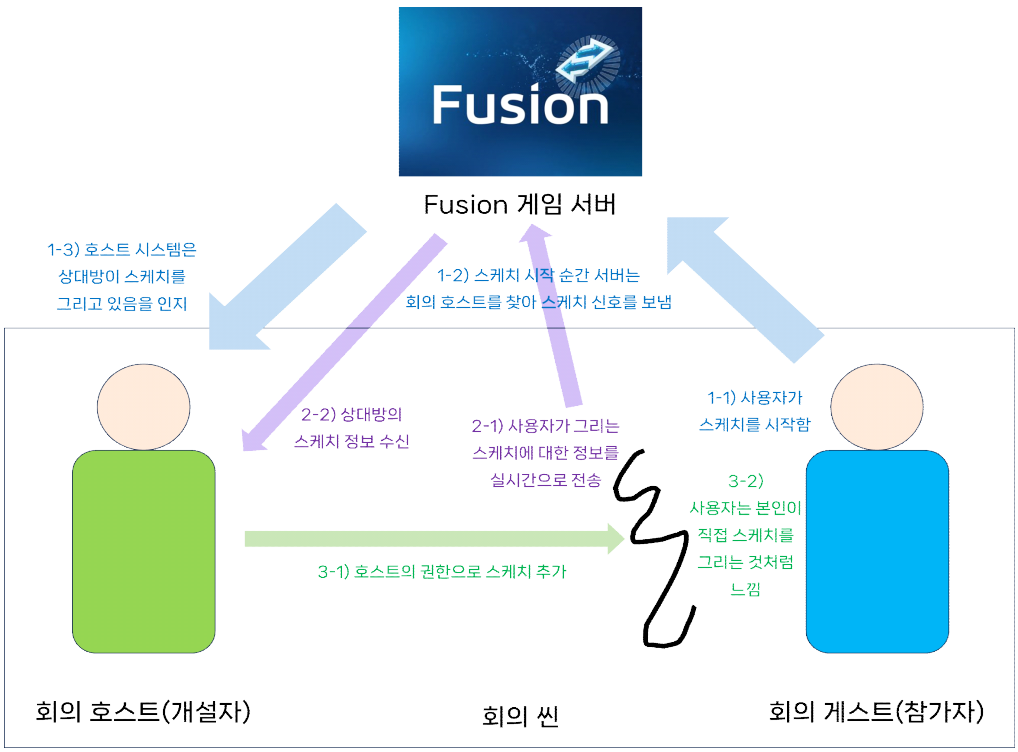

그림 21. Photon Fusion에서 RPC를 활용한 스케치 공유 예시

이러한 문제점과 구현의 난이도로 인해, VR 회의장에서는 스케치를 한붓그리기 수준으로 그리는 것만 허용하였다. 또한 VR 회의에서의 스케치는 정중앙에 그리는 경우가 많아 특별히 이동시킬 필요가 없다는 것도 고려되었다.

3.3.7 웹 및 데이터페이스 구현

현대 사회의 업무에 있어서 컴퓨터는 필수불가결의 요소다. 그래서 WebRTC 기술을 이용하여 내 PC의 화면을 가상/증강 현실 공간에 표시해주는 시스템을 구상했다. 실시간 스트리밍 기술로 Appls HLS, DASH, RTP 등 많은 프로토콜이 존재하지만, 그 중 WebRTC를 선정한 이유는 다음과 같다.

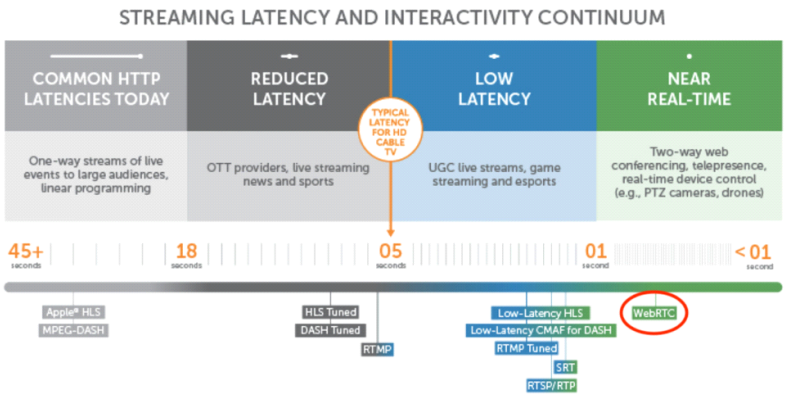

그림 22. 영상 전송 프로토콜 지연 시간 비교 그래프,
WebRTC가 거의 실시간에 가까운 지연 시간을 보인다.

먼저 첫 번째는 낮은 지연 시간이다. 마우스나 키보드를 이용한 상호작용이 마치 실제 PC를 사용하듯 원활하게 이루어져야 하기 때문에 낮은 지연 시간이 기술 선정에 있어 가장 중요했다. WebRTC는 많은 영상 송수신 프로토콜 중 가장 낮은 지연 시간을 보였다.

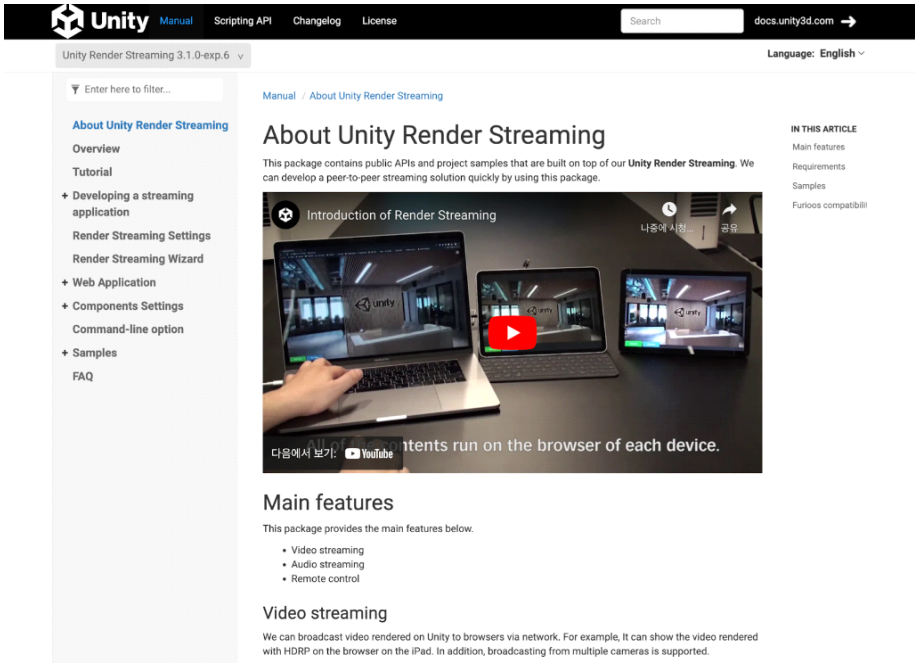

그림 23. Unity에서 제공하는 WebRTC 스트리밍 패키지

두 번째는 메인 개발 툴인 Unity의 전용 개발 패키지 존재이다. WebRTC는 브라우저(e.g. Chrome, Firefox, Edge) 간의 데이터를 송수신하는 기술이기 때문에 Unity에서 브라우저의 역할을 해줄 패키지가 필요했다. 더불어 Unity에서 제공하는 패키지의 샘플을 활용함으로써 구현 난이도 또한 낮출 수 있었다.

구현을 위한 메인 로직은 다음과 같다.
구현은 Unity Render Streaming 패키지에 포함되어 있는 샘플을 활용하였다. 해당 샘플의 내용으로 Receiver, Broadcast, Bidirectional 등 여러 상황에 맞는 샘플이 존재했지만, 이 프로젝트의 상황에 맞게 브라우저에서 처리한 영상의 내용을 전송해주는 BIdirectional 샘플을 활용하였다.

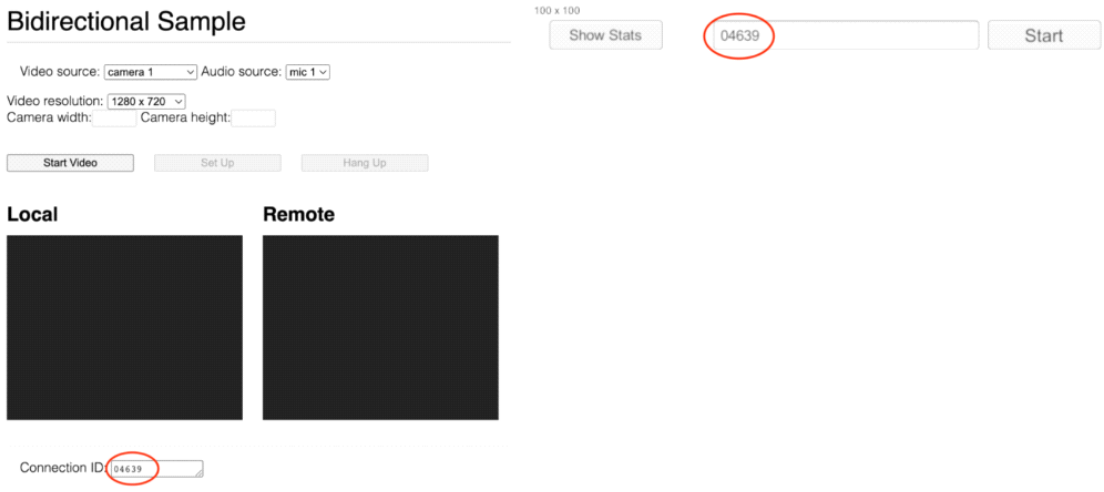

그림 24. 왼쪽: PC, 오른쪽: Unity. Connection ID를 일치시켜야 영상이 전송된다.

Bidirectional 샘플은 Unity 내에서 영상을 송수신할 수 있는 GameObject와 웹에서 영상을 송수신할 수 있는 Web Application으로 구성되어 있다. 아래 사진과 같이 같은 네트워크 안에서 Connection ID를 같은 값으로 맞춰주면 PC로부터 송신받은 영상을 Unity에서 수신할 수 있는 방식이다.

```
async startLocalVideoScreen() {
  try {
    const constraints = {
      // PC 전체 화면 기본 선택
      video: {
        displaySurface: 'monitor'
      },
      audio: true
    };
    const localStream = await navigator.mediaDevices.getDisplayMedia(constraints);
    this.localVideo.srcObject = localStream;
    await this.localVideo.play();
  } catch (err) {
    Logger.error(`mediaDevice.getDisplayMedia() error:${err}`);
  }
}
```

그림 25. 웹 표준 API getDisplayMedia() 사용 예시

기존 샘플의 내용은 PC에 연결된 웹캠의 영상을 전송했지만, 본 프로젝트에서는 PC의 화면을 Unity로 전송해줘야 하기 때문에 웹 표준 API인 getDisplayMedia()를 활용했다.

https 통신을 위한 SSL 인증서 발급 (OpenSSL) 내용은 다음과 같다.

그림 26. 구현 당시 발급받은 SSL 인증서

사용자의 카메라나 마이크를 스트리밍하는 getUserMedia()나 PC 화면을 스트리밍하는 getDisplayMedia()와 같이 상당한 개인 정보 보호 문제를 수반하는 API들은 https와 같은 보안 환경에서만 정상적으로 작동하게 설계되어 있다. 그래서 OpenSSL을 활용해 인증키를 발급 받아 https 환경에서 서버가 구동되도록 구현하였다.

본 연구에서 개발된 시스템의 회원가입 및 로그인 기능 내용은 다음과 같다. 이는 사용자의 편의성과 보안을 중점적으로 고려하여 구현되었다. 회원가입 과정에서 사용자는 아이디, 이름, 비밀번호를 입력하며, 이때 입력된 정보는 유효성 검사를 거친다. 이 과정은 EJS 파일과 라우터를 통해 구현되었으며, 유효성 검사는 POST 방식으로 데이터가 전송되기 전에 이루어진다. 특히, MySQL 데이터베이스를 참조하여 이미 존재하는 아이디(Primary Key)인 경우 오류 메시지를 반환하고, 그렇지 않은 경우 회원가입을 완료하도록 설계되었다.
로그인 기능의 경우, 역시 EJS 파일과 라우터를 사용하여 구현되었다. 사용자가 입력한 로그인 데이터는 POST 방식으로 라우터에 전달되며, 여기서 데이터베이스에 저장된 정보와 비교 처리된다. 로그인에 성공한 사용자에 대해서는 cookieparser를 활용하여 현재 시간으로부터 900000ms 동안 유효한 쿠키를 생성하여, 사용자의 이름을 화면에 표시할 수 있도록 하였다. 이 쿠키는 로그아웃 시 삭제되도록 설정되어 있다.

```
class UserStorage {
  static getUserInfo(id) {
    return new Promise((resolve, reject) => {
      const query = "SELECT * FROM users WHERE ID = ?;";
      db.query(query, [id], (err, data) => {
        if (err) reject(err);
        resolve(data[0]);
      });
    });
  }

  static async save(userInfo) {
    return new Promise((resolve, reject) => {
      const query = "INSERT INTO users(id, name, pw) VALUES(?,?,?);";
      db.query(query, [userInfo.id, userInfo.name, userInfo.pw], (err) => {
        if (err) reject(`이미 존재하는 아이디입니다. `);
        resolve({ success: true });
      });
    });
  }
}
```

```
class User {
  constructor(body) {
    this.body = body;
  }

  async login() {
    const client = this.body;
    try {
      const user = await UserStorage.getUserInfo(client.id);
      if (user) {
        if (user.id === client.id && user.pw === client.pw) {
          return {
            success: true,
            name: user.name,
            connectoinid: user.connectionID,
          };
        }
        return { success: false, msg: "비밀번호가 틀렸습니다." };
      }
      return { success: false, msg: "아이디가 존재하지 않습니다." };
    } catch (err) {
      return { success: false, msg: err };
    }
  }

  async register() {
    const client = this.body;
    try {
      const response = await UserStorage.save(client);
      return response;
    } catch (err) {
      return { success: false, msg: err };
    }
  }
}
```

그림 27. 왼쪽: 회원가입 구현 코드 일부, 오른쪽: 로그인 구현 코드 일부.

3.3.8 AR/VR 및 웹 통합

본 연구에서 개발된 시스템의 통합 및 테스트를 위해서는 가장 먼저 웹 서버의 가동이 필수적이다. 이는 시스템의 핵심적인 부분인 유저 데이터베이스와 WebRTC 체계를 포함하고 있기 때문이다. 웹 서버가 작동하게 되면, Unity 애플리케이션을 실행할 수 있으며, 사용자가 Unity 앱과 웹 앱에 로그인함으로써 시스템의 사용 준비가 완료된다.

본 시스템의 특징 중 하나는 로컬 웹 서버와 Photon Fusion이 분리되어 구현되어 있다는 점이다. 이러한 이원화된 구조를 자연스럽게 통합하는 것이 개발 과정에서의 주요한 과제였다. 결과적으로, XR 애플리케이션은 Photon Fusion 게임 서버와 로컬 웹 서버, 이 두 서버로부터 정보를 주고받게 설계되었다. AR 모드에서는 로컬 웹 서버와 연동된 데이터베이스로부터 사용자 정보를 얻어내며, WebRTC를 통해 영상을 공유받는다. 반면, VR 모드에서는 Fusion 서버를 사용하여 회의실을 생성하고 사용자의 상태를 공유하며, 영상 공유에는 WebRTC가 사용된다.

이러한 방식으로 본 시스템은 다양한 기능과 모드를 효율적으로 통합하며, 사용자에게 원활한 경험을 제공한다.

3.4 구현 결과

3.4.1 사용자 요구사항 구현 정도

사용자 요구사항(표 1)을 기준으로 구현 정도를 백분율(%)로 나타내었을 때, 다음과 같이 언급한 요구사항을 제외한 요구사항은 구현 완료(100%)하였다. 특히, 필수 사항은 대부분 구현 완료하였으며, 선택 사항은 부분적으로 구현

하였다. 현실을 고려한 가상 물체 배치(AR-6)은 0%, 다양한 방 설정 저장(AR-7)은 20%, AR Workspace와의 연동(VR-3)은 90%, 회의 초대 기능 구현(VR-7)은 0%, 가상 물체의 속성 수정(XR-4)는 90%, 키보드로 메모 작성(XR-5)은 30%, AR/VR 전환(XR-6)은 0% 구현하였다.

3.4.2 구현 결과 사진

그림 28. 왼쪽: AR 환경에서 PC 공유, 오른쪽: AR 환경에서 메모 작성

사용자는 XR 애플리케이션에 접속하게 되면, 단일 사용자를 위한 AR 업무 환경 관리 시스템에 접속된다. 그림 28처럼 개인 업무에 필요한 메모, 스케치, PC 화면 등을 현실 공간에 증강하고, 위에서 제시한 인터페이스를 활용하여 원거리 배치 및 조작을 할 수 있는 시스템이다.

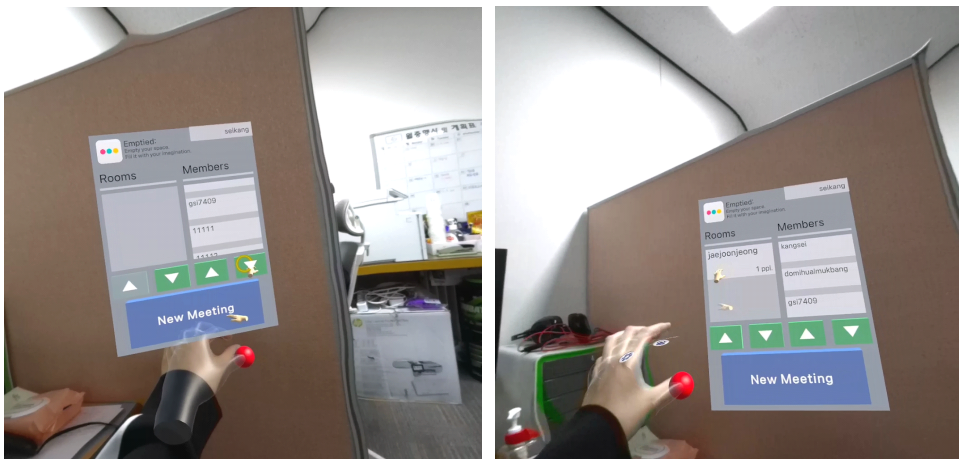

그림 29. 왼쪽: 웹에서 회원가입한 사용자 목록, 오른쪽: 다른 사용자가 생성한 회의방에 접속하는 모습

본 연구에서 개발된 VR 회의 시스템은 개인 업무 수행 중에도 손쉽게 회의

로 전환할 수 있는 기능을 제공한다. 사용자가 회의를 시작하고자 할 때는 'New Meeting' 버튼을 눌러 새로운 회의방을 생성할 수 있다. 또한, 이미 다른 사용자가 생성한 회의방에 접속하는 것도 가능하다(그림 29 오른쪽 참고). 이 과정에서 사용자는 웹 사이트에서 회원가입을 진행한 사용자들이며, 회의방 옆에 표시된 'Members'를 통해 참가자들의 사용자 아이디를 확인할 수 있다(그림 29 왼쪽 참고). 이러한 시스템의 설계는 사용자가 VR 환경 내에서 효율적으로 회의를 관리하고 참여할 수 있도록 하여, 업무 수행과 회의 참여 간의 원활한 전환을 지원한다.

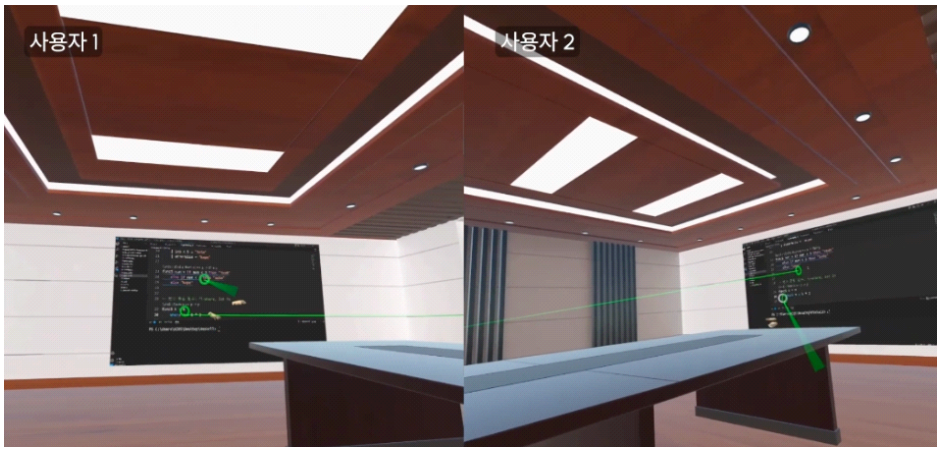

그림 30. 두 사용자의 VR 회의 Pair Programming 시나리오 (시선 포인터 공유)

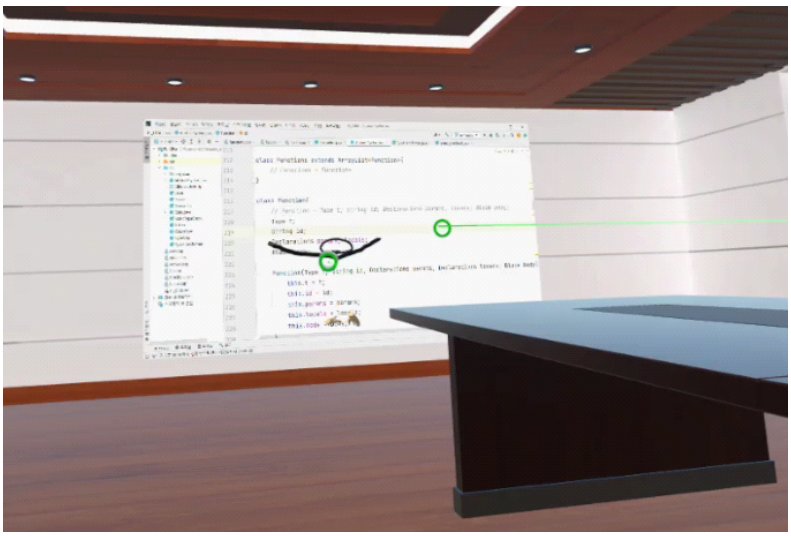

그림 31. 두 사용자의 VR 회의 Pair Programming 시나리오 (시선 포인터 및 스케치 공유)

VR 회의 시스템은 가상 객체를 공유하며 협업할 수 있는 시스템으로, 위에서 제시한 인터페이스로 복수의 사용자 함께 가상 물체 및 메모 추가, 3D 스케치 등을 조작 및 공유할 수 있다(그림 31 참고). 또한, 인터페이스를 통해 그림 30과 그림 31처럼 시선 포인터도 공유가 가능하다. AR 업무 환경 시스템과 VR 회의 시스템을 통합하여 AR 환경에 추가된 객체를 협업 시스템으로 불러와 팀 회의에 활용할 수 있다. 또한 WebRTC를 사용하여 사용자 간 통신과 PC 화면 공유 기능을 제공한다.

4. 실험 및 분석

4.1 실험 개요

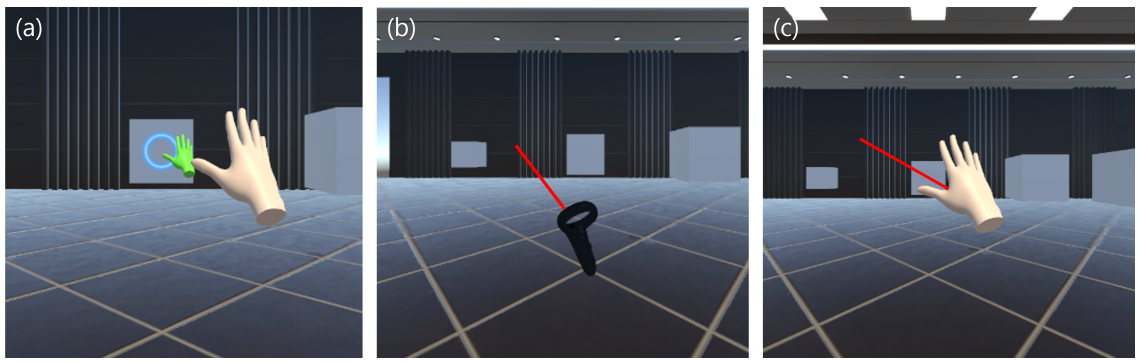

그림 32. 본 실험에서 사용한 인터페이스 (a) 눈+손 인터페이스 (b) Controller (c)Hand

본 연구에서는 “눈+손 인터페이스”의 성능을 평가하기 위해 사용자 실험을 실시하였다. 이 실험에는 총 21명이 참여하였으나, 시선 추적의 불확실성 등의 문제로 3명은 분석에서 제외되었다. 실험은 눈+손 인터페이스(그림 32 a 참고)와 비교 대상으로 Controller(그림 32 b 참고) 및 Hand(그림 32 c 참고) 인터페이스를 사용하였다. 이들 Controller와 Hand 인터페이스는 Meta Quest 기기에서 지원하는 기본 인터페이스로, Spatial과 Meta Horizon Workrooms 등의 VR/AR 애플리케이션에서 널리 사용된다. 실험에서는 두 가지 주요 Task를 수행하도록 하였다. 첫 번째 Task는 좌우 방향의 물체를 선택하는 것이고, 두 번째 Task는 앞뒤 방향의 물체를 조작하는 것이다. 각 실험마다 소요된 총 시간과 시스템 사용성 평가(SUS) [6]를 측정하였다. 이를 통해 눈+손 인터페이스의 성능과 사용성을 객관적으로 평가하고, 기존 인터페이스와의 비교를 통해 그 효과성을 검증하였다.

4.2 실험 방법

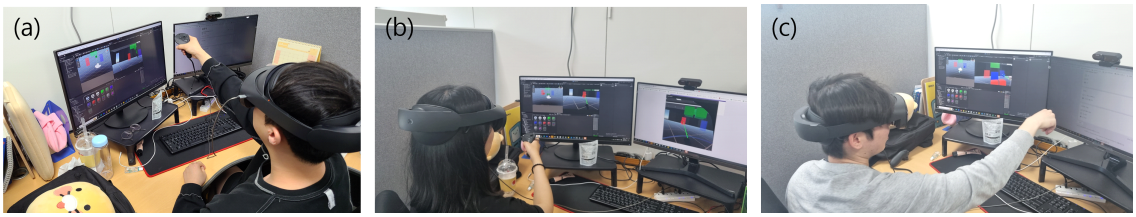

그림 33. 실험 전 각 인터페이스에 익숙해지기 위한 Task를 수행하는 참가자 모습

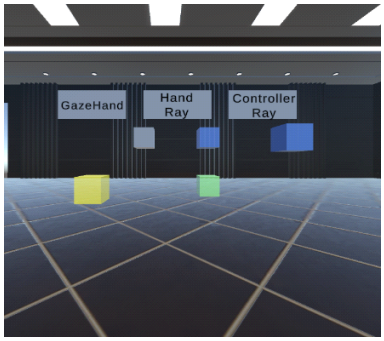

그림 34. 실험 전 Task

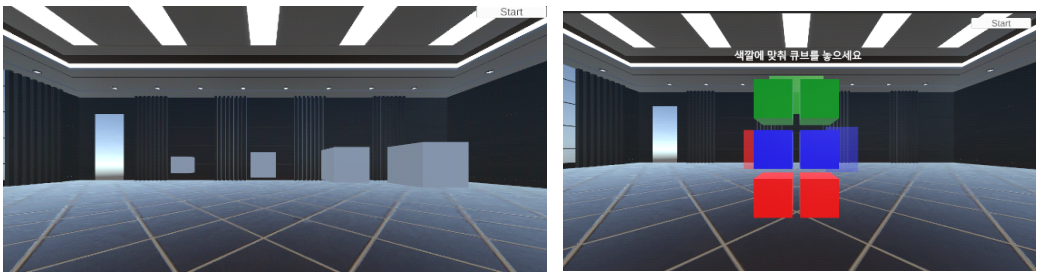

그림 35. Task 사진. 왼쪽: 첫 번째 Task(좌우 방향 물체 선택), 오른쪽: 두 번째 Task(앞뒤 방향 물체 조작)

실험 전 참가자들은 Meta Quest Pro 시선 조정을 통해 눈+손 인터페이스를 사용하기 위한 준비를 한다. 이후, 각 인터페이스에 대한 설명을 들은 후, 인터페이스에 익숙해지기 위해 참가자는 자유롭게 가상 오브젝트를 조작한다. (그림 34 참고) 인터페이스에 익숙해졌다면, Task를 진행한다. 첫 번째 Task는 좌우 물체 선택으로, 크기가 다른 4가지 Cube를 최대한 빠르게 선택하는 것이다(그림 35 왼쪽 참고). 각 Cube 크기는 5 ~ 20cm, 즉, 5cm 단위로 차이가 나며 각 Cube는 1m씩 떨어져 있다. 또한 모든 Cube는 참가자로부터 2m 떨어져 있다. 첫 번째 Task는 각 인터페이스별로 총 3번 진행하며 한 번의 실험이 끝날 때마다 총시간이 CSV 파일로 저장되고, 참가자는

SUS 설문지를 작성한다. 두 번째 Task는 앞뒤 방향의 물체 조작으로, 크기가 같고, 색깔이 다른 Cube 6개를 색깔에 맞추어 판에 넣는 것이다(그림 35 오른쪽 참고). 각 Cube 크기는 25cm이며 참가자로부터 1m 떨어져 있다. 색깔판은 참가자로부터 2m, 2.5m, 3m 떨어져 있고, 참가자는 Cube를 잡고 앞으로 물체를 옮긴다. 마찬가지로 Task는 3번 진행하며 총시간과 SUS 설문지 데이터를 측정한다.

4.3 실험 결과

본 연구에서 수행된 실험 결과의 데이터는 CSV 파일로 저장되었으며, SPSS 통계 분석 프로그램을 사용하여 정규성 및 유의성을 검증하였다. 실험 측정값은 각 Task마다 걸린 시간과 SUS 설문지 데이터이다. 정규성 검정을 위해 Shapiro-Wilk test가 사용되었는데, 이는 Shapiro와 Wilk가 1965년에 발표한 방법으로, 데이터 셋의 정규 분포를 검정하는 데 적합하다. 또한, 비모수 자료의 경우 Wilcoxon Signed test를 적용하여 단일 집단 내에서의 차이를 검정하였다. 이는 정규분포를 따르지 않는 자료에 대해 사용되는 비모수 통계 방법이다. 반면에 모수 자료에 대해서는 Paired-t test를 사용하여 정규분포를 따르는 데이터 집단 간의 차이를 검정하였다. 유의성 계산에서는 전통적인 가설검정 방식을 따라, p-값이 0.05 이하일 경우를 통계적으로 유의미한 차이가 있는 것으로 간주하였다. 이러한 통계적 방법론을 통해 실험 결과의 신뢰성을 검증하고자 하였다.

4.3.1 시간

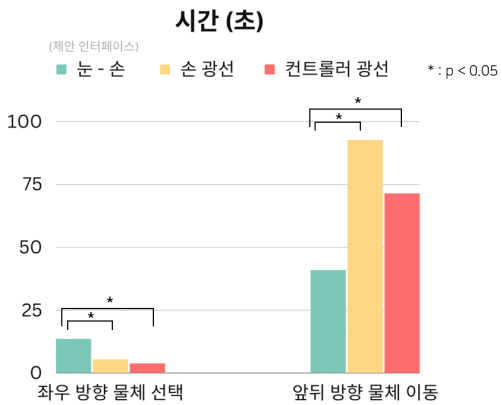

그림 36. 시간에 대한 결과 그래프
(*은 p<0.05인 것을 나타냄)

첫 번째 Task에서 Shapiro-Wilk test를 통해 정규성을 검정하고, p < 0.05일 경우 모집단이 정규분포를 따르지 않는다고 가정한다. 분석 결과 눈+손 인터페이스(M=13.41, SD=9.16, p<.001)와 Controller(M=3.72, SD=1.85, p<.001)는 정규성을 만족하지 않는 것으로 나타났으나, Hand(M=5.35, SD=1.82, p=.163)는 정규성을 만족하는 것으로 관찰되었다. 이에 따라 Wilcoxon Signed test를 진행하였고, 그 결과 눈+손 인터페이스-Controller, 눈+손 인터페이스-Hand에서 p<.001이 나와, 이들 간에 통계적으로 유의미한 차이가 있는 것으로 확인되었다.
두 번째 Task에서도 마찬가지로 Shapiro-Wilk test를 통해 정규성을 검정하였다. 그 결과 눈+손 인터페이스(M=40.83, SD=15.58, p=.085), Hand(M=92.56, SD=59.49, p=.358), Controller(M=71.38, SD=29.11, p=.603) 모두 인터페이스가 정규성을 만족한다고 볼 수 있다. 이에 따라, Paired-t test를 진행하였고, 첫 번째 Task와 비교를 위해 Wilcoxon Signed test도 진행하였다. Paired-t test 결과로는 눈+손 인터페이스-Controller(p=.001)와 눈+손 인터페이스-Hand(p<.001)가 p<.05를 만족하므로 통계적으로 유의미한 차이가 있다. Wilcoxon Signed test 결과로는 눈+손 인터페이스-Controller(p=.002)와 눈+손 인터페이스-Hand(p<.001)가 p<.05를 만족하므로 통계적으로 유의미한 차이가 있다.

4.3.2 사용성(SUS)

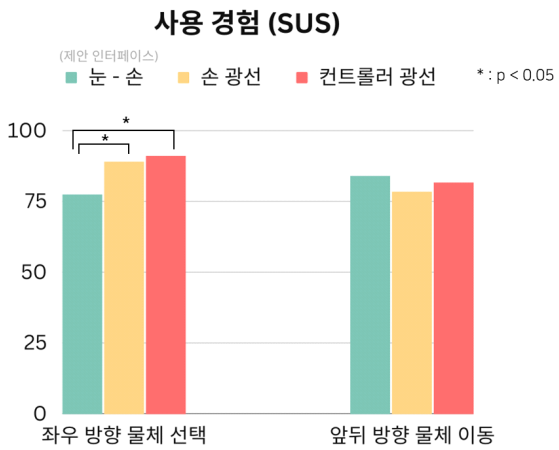

그림 37. 사용성에 대한 결과 그래프
(*은 p<0.05인 것을 나타냄)

본 연구에서는 SUS(System Usability Scale Test)를 활용하여 각 인터페이스의 시스템 사용성을 평가하였다. SUS는 시스템의 유용성을 척도로 하여,

점수를 계산함으로써 사용성의 존재 여부 및 수준을 파악할 수 있게 해준다. 첫 번째 Task에서 눈+손 인터페이스(M=77.36, SD=18.38), Hand(M=88.88, SD=10.85), Controller(M=90.97, SD=9.96) 모두 SUS 점수가 68점 이상이므로[8] 사용성을 가지되, Controller와 Hand는 높은 사용성을, 눈+손 인터페이스는 상대적으로 낮은 사용성을 가지는 것을 확인할 수 있었다. 또한 SUS와 같은 설문지를 사용하여 정수형 값을 얻기 위해서는 Wilcoxon Signed test를 통해 유의성을 검정해야 한다. 그 결과, 눈+손 인터페이스-Controller(p=.002)와 눈+손 인터페이스-Hand(p=.004)에서 p<.05이므로 통계적으로 유의미한 차이가 있다.
두 번째 Task에서 눈+손 인터페이스(M=83.88, SD=13.61), Hand(M=78.33, SD=16.22), Controller(M=81.52, SD=15.17) 모두 SUS 점수가 68점 이상이므로 사용성을 가지되, 눈+손 인터페이스에서 가장 높은 점수를 보였다. Wilcoxon Signed test를 통해 유의성을 검정한 결과, 눈+손 인터페이스-Controller(p=.717)와 눈+손 인터페이스-Hand(p=.263)에서 p>.05이므로 통계적으로 유의미한 차이가 없다.

4.3.2 사용자 인터뷰

모든 Task의 실험이 끝난 후 “눈+손 인터페이스”에 대한 의견을 자유롭게 말하는 사용자 인터뷰를 진행하였다.

참여자	인터뷰 내용
P2	손으로 도구를 잡거나 손을 들을 필요가 없어서 편했고, 좌우상하 움직임도 편했다.
P4	고개를 돌리면 보이는 화면이 바뀌고, 내 손 모션을 통해 특정 작업을 할 수 있어서, 내가 실제로 가상 세계에 들어온 것 같았다. 조작도 직관적이고 편리해서 이것을 활용한 시스템이 나온다면 사용해볼 것 같다.
P5	확실히 팔을 여러 번 움직이며 물체를 옮기는 것보다는 빨랐다. 하지만 눈으로 이동 시 내가 원하는 만큼 이동이 되는지는 잘 모르겠다. 그 점이 보완된다면 게임으로든 비즈니스로든 여러 방면에서 활용하기 좋을 것 같다.
P6	눈으로 물체 이동시키는 게 컨트롤러나 손보다는 훨씬 편했다! 빠르기도 하고 팔 덜 움직여도 되고 물체가 시선 따라서 잘 움직이는 게 되게 신기했다.
P7	눈동자를 이용하여 물체를 옮길 수 있어 재밌는 경험이었다. 팔을 이용해서 움직이는 것보다 빠르고 사람들이 더 쉽게 사용할 수 있을 것 같다!
P8	움직이지 않고 물체를 뒤쪽으로 옮기는 테스트에서 다른 방법과 대비되게 눈+손 인터페이스의 편리함을 느낄 수 있었다. 다른 방법들은 오브젝트 클릭 후 앞으로 손을 뻗는 과정을 반복하다 보니 팔이 조금 아팠는데 눈+손 인터페이스는 눈을 조금 움직여도 금방 먼 초점으로 이동해서 편리했다. 하지만, 시선 초점에서 손이 인식되는 것이 좀 더 정확해지면 좋을 것 같다.

표 3. 사용자 인터뷰 결과

5. 결론 및 향후과제

5.1 논의

본 프로젝트는 크게 3개 단계로 나누어 수행되었고, 각 단계별로 주목하거나 논의할 만한 사항을 정리했다.

5.1.1 AR 및 VR 환경에서 본 시스템을 사용하기 위한 사용자 인터페이스와 상호작용 인터페이스를 정의하는 단계

인간-사용자 상호작용(HCI)의 중요한 요소로는 유용성, 사용성, 감성이 있는데, 전통적으로는 유용성이, 근대에는 사용성이, 최근에는 감성이 중시되고 있다. 본 인터페이스는 3가지 요소 중 유용성을 가장 추구했다고 생각한다. 하지만 사용성 측면의 경우 아쉬운 점이 있었다. 그 이유로는 VR 기기에서 제공하는 눈 트래킹 정보의 정확도 때문에 눈 포인터가 다소 떨리는 문제가 있어 실제 사용 시 집중해야 했기 때문이다. 이를 향상하려면 눈 트래킹 보정 알고리즘을 적용해야하지만, 실제로 적용하진 못했다. 또한 손가락을 맞닿아 선택하는 인터페이스 역시 검지(OK 제스처) 이외의 손가락은 익숙하지 않아 실제 사용 시 어색함을 느낄 수도 있다.

5.1.2 AR 및 VR 시스템과 이를 보조하기 위한 Web 시스템을 개발하는 단계

본 프로젝트의 핵심 영역이자 최대 애로사항으로, 우선 현실적으로 AR/VR 개발은 지금까지 비주류 영역으로 참고할 자료가 다른 플랫폼(PC, Mobile)에 비해 부족하여 개발의 많은 부분을 직접 고민하고 추가해야 했다. 거기에 더해 WebRTC는 웹 기반으로 동작하는데 이를 Unity라는 다른 개발 플랫폼과 연동하는 일 역시 어려웠다. 다행히 샘플 패키지를 찾아 많은 부분을 해결할 수 있었지만 소스 코드를 수정할 수는 없어 불안정한 동작을 해결하기는 쉽지 않았다. 예를 들어 WebRTC 코드가 실행되는 중에 VR 회의를 종료하면 해당 코드가 종료될 때까지 VR 회의 화면이 그대로 멈추는 증상이 있었다. 또한 다인용 시스템으로서 네트워크를 사용하는 만큼 프로젝트를 테스트하는 것 역시 어려웠다.

5.1.3 인터페이스의 성능을 평가하는 사용자 실험 단계

인공지능 기술을 이용해 컴퓨터로 프로그램만 짜면 되는 인공지능 영역의 성능 실험과 달리, AR/VR 시스템을 포함한 사용자 상호작용 분야에서는 실제 사람이 해당 시스템을 실제로 사용하여 특정 업무를 완수하고 그에 대한 객

관적 지표와 주관적 평가를 통해 성능을 측정한다. 이런 실험이 의미가 있으려면 실험 참가자의 유형이 다양해야 한다. 예를 들어 연령대, 성별, 안경 착용 여부, AR 및 VR 시스템의 사용 경험 등을 고려하는 것이 좋다. 그러나 참가자 모집에 현실적인 어려움이 있어 본 연구에서는 또래 학생들 위주로만 실험을 진행했다는 아쉬운 점이 있다. 특히 처음 본 시스템을 사용해본 사람들은 본 인터페이스를 다소 낯설어하고 적응하는 데에 시간이 다소 걸렸다는 점에서 본 인터페이스를 익히는 데에 어려움이 있음을 알 수 있었다.

5.2 한계점 및 추후 연구

본 연구에서는 현재 개발된 시스템의 한계점과 추후 연구 방향에 대해 고찰한다.

첫째, 현재 본 시스템은 회의실의 최대 인원을 4명으로 제한하고 있는데, 그 이유는 WebRTC를 사용했기 때문이다. WebRTC를 이용해 영상을 공유할 때는 PC 소유자가 다른 참가자와 모두 1:1의 연결을 맺어야 작동하는 구조이다. 그러므로 이론적으로는 인원을 더 늘릴 수 있으나 사용자별 대역폭의 한계가 있기 때문에 한정 없이 늘릴 수는 없다. 이 문제를 해결하기 위해서는 WebRTC 연결을 N:M 구조로 수정 확장하거나 별도의 서버에서 이 작업을 하게 해야 한다. 또는 다른 스트림 솔루션을 사용해야 한다.

둘째, 본 시스템에 필요한 서버가 유저 데이터베이스와 WebRTC에 관여하는 local 웹 서버와 VR 회의 및 유저 정보 공유에 관여하는 Fusion 서버로 이원화되었다는 문제가 있다. 이는 Fusion 측에서 영상 공유 API를 제공하지 않기 때문이다. 하지만 이런 서비스를 제공하기 어려운 이유는 유저의 움직임, 채팅, 음성 데이터는 필요한 대역폭이 크지 않지만, 영상 송출은 많은 대역폭이 필요하여 서버가 부담되기 때문이다. 이 부분은 양쪽 시스템을 모두 구현하지 않고는 해결하기 어려운 현실적인 한계점으로 꼽힌다.

셋째, 요구사항에 명시했던 사용자 회의 초대 기능은 현재 구현하지 못했다. 이를 구현하려면 유저 데이터베이스를 확장하고 알림 시스템을 별도로 구축해야 하는데, 시간 및 우선순위에 따라 생략했다.

넷째, 현재 VR 회의장에서는 AR 모드와 달리 오브젝트 수정 기능이 구현되지 않았다. 물론 회의 특성상 속성을 바꾸는 사례가 특별히 많다고 생각하지 않아 사용성에는 큰 문제가 없으나, 완성도 측면에서 보충되면 좋아질 부분이다.

다섯째, Quest Pro는 카메라 영상 기반으로 AR을 지원하고, 해당 영상은 RGB 카메라 기반이 아닌 트래킹을 위한 적외선 카메라에 색상을 입힌 형태라 실제 착용 시 현실의 모습이 비교적 흐릿하게 출력되어 다소 현실감이 떨

어진다는 단점이 있다. 따라서 RGB 카메라 기반 AR 또는 See-through AR을 지원하는 기기에 적용하면 사용자 경험이 향상될 것으로 기대된다.
마지막으로, 현재 본 인터페이스의 가장 큰 단점은 시선 거리 조절이 힘들다는 점인데, 이는 무조건 자신의 손을 최초 충돌 점의 끝부분에 갖다두기 때문이다. 다시 말하면, 공중에 떠 있는 물체 옆으로 다른 물체를 옮길 수는 있지만, 아무것도 없는 빈 공간의 한가운데에 물체를 갖다 두기에는 매우 불편하다. 따라서 시선 거리를 원하는 때에 원하는 만큼 조절할 수 있는 방법이 추후 연구 및 적용되어야 한다.

5.3 결론

본 연구에서는 사용자 간의 협업을 위해 가상현실 기기의 시선 정보를 활용한 AR/VR 애플리케이션을 설계하고, 그 과정에 필요한 인터페이스들을 정의하고 요구사항에 따라 구현하였다. 그리고 인터페이스의 성능 평가를 위한 사용자 실험을 진행하였다. 본 연구 프로젝트 수행을 통해 현실의 협업 문제를 해결하기 위한 AR/VR 시스템의 활용 가능성을 확인하고, 추후 더 발전시킨다면 사람들의 업무 능력 향상 및 AR/VR 산업 발전에 기여할 수 있을 것으로 기대한다.

6. 참고문헌

[1] I. Poupyrev, M. Billinghurst, S. Weghorst, and T. Ichikawa, The go-go Interaction Technique: Non-Linear Mapping for Direct Manipulation in VR. Proceeding of UIST, 1996, ACM, New York, NY, USA, pp. 79-80.

[2] K. Pfeuffer, B. Mayer, D. Mardanbegi, and H. Gellersen, Gaze + Pinch Interaction in Virtual Reality. Proceeding of. SUI, 2017. ACM, New York, NY, USA, pp. 99-108.

[3] C. Liu, A. Plopski, and J. Orlosky, Orthogaze: Gaze-based Three-Dimensional Object Manipulation Using Orthogonal Planes. Computers & Graphics, Vol. 89, No. 1, pp. 1-10.

[4] D. Yu, X. Lu, R. Shi, H. Liang, T. Dingler, E. Velloso, and J. Goncalves, Gaze-Supported 3D Object Manipulation in Virtual Reality. Proceeding of CHI, 2021, ACM, New York, NY, USA, Article 734, pp.1-13.

[5] K. Ryu, J. Lee, and J. Park, GG Interaction: A Gaze-Grasp Pose Interaction for 3D Virtual Object Selection. Journal on Multimodal User Interfaces Vol. 13, No. 4, pp. 383-393. Dec 2019.

[6] J. Brooke. SUS: A Quick and Dirty Usability Scale. In Usability Evaluation in Industry, 1st ed., CRC Press, Boca Raton, Florida, USA, 1996, pp. 189-194.

[7] D. A. Bowman and L. F. Hodges. An Evaluation of Techniques for Grabbing and Manipulating Remote Objects in Immersive Virtual Environments. Proceeding of I3D, 1997, ACM, New York, NY, USA, pp. 35-38.

[8] James R. Lewis and Jeff Sauro. 2018. Item benchmarks for the system usability scale. J. Usability Studies 13, 3 (May 2018), 158-167.

[9] HQ ArchVis Conference Room 1.1
https://assetstore.unity.com/packages/3d/environments/hq-archviz-conference-room-127663

[10] Avatar Full body - Ready Player Me; pmariano
https://sketchfab.com/3d-models/avatar-full-body-ready-player-me-pmariano-a9c1f5d2cd7c4ca3bb46272998