Pusan National University
Computer Science and Engineering
Technical Report 2024-10

# VR 기반 재난 상황 교육 및 예방 시뮬레이션 시스템 구현

저자 1 201824559 이종민

저자 2 201924559 장승우

저자 3 202055534 노윤정

지도교수 이명호

# 목 차

# 1. 서론

## 1.1. 과제 배경

재난은 언제나 예측할 수 없고, 실제 상황에서 적절한 대처를 하는 것은 쉽지 않다. 이에 따라, 재난 발생 시 효과적으로 대응하는 능력을 길러주는 교육이 필수적이다. 특히 화재와 같은 긴급 상황에서의 대처 능력은 생명을 구하는 중요한 요소로 작용한다. 전통적인 교육 방식은 주로 이론적인 교육에 치우쳐 있어 실제 상황에 적응하는 데 한계가 있다. 이를 보완하기 위해 최근에는 가상현실(VR) 기술을 이용한 교육이 주목받고 있다. VR 기술은 사용자가 실제와 유사한 재난 상황을 체험할 수 있게 함으로써, 안전 대처 능력을 더욱 효과적으로 향상하는데 기여할 수 있다.

## 1.2. 기존 문제점

기존의 재난 교육은 주로 영상이나 이론 중심의 교육이었고, 실제 상황을 경험하기 어려웠다. 또한, 제한된 자원과 환경에서 모든 체험자가 개별적으로 실습할 수 없다는 문제도 있었다. 실질적인 재난 대응 능력을 기르기 위해서는 현장감 있는 학습이 필요하지만, 기존 방법으로는 이를 충족하기 어려웠다. 이러한 점은 재난 발생 시 적절한 대응이 어려운 주요 원인으로 작용한다.

더욱이 사전 조사에 따르면, VR 기술을 활용한 재난 상황 시뮬레이션 연구는 많이 존재하지만, 체험자의 행동을 실시간으로 모니터링하고 1:1로 교육할 수 있는 감독관 시스템을 도입한 연구는 찾아볼 수 없었다. 대부분의 기존 연구는 VR 기술을 통한 체험자의 몰입과 경험에 초점을 맞추고 있지만, 실시간 피드백이나 개별 지도와 같은 기능을 제공하지 않는다.

이러한 점에서 본 과제는 체험자의 행동을 감독관이 실시간으로 모니터링하고 즉각적인 피드백을 제공하는 시스템을 개발함으로써, 보다 효과적이고 맞춤형 교육을 제공할 수 있다는 강점이 있다. 이는 기존의 단순한 시뮬레이션 방식과 차별화되며, 재난 대응 교육의 질을 크게 향상할 것으로 기대된다.

### 1.3. 과제 목표

본 과제의 필요성은 **체험자와 감독관 간의 실시간 상호작용**을 통해 맞춤형 재난 교육을 제공하는 데 있다. 감독관이 체험자의 행동을 실시간으로 모니터링하고, 필요할 때 즉각적인 피드백을 제공함으로써, 기존 교육의 한계를 넘어서는 보다 **효과적인 재난 대응 교육**을 구현할 수 있다. 이러한 시스템은 체험자가 재난 상황에서 적절한 행동을 즉각적으로 습득할 수 있도록 도와줄 뿐만 아니라, **실질적인 대응 능력을 향상**시키는 데 기여할 것이다.

## 2. 시스템 요구조건 및 설계

### 2.1. 시스템 요구 조건

본 과제는 유니티3D와 VR 기술을 활용하여 재난 상황을 실감 나게 재현하는 시뮬레이션 시스템으로 교육과 예방을 목표로 한다. 체험자가 VR 기기를 PC에 연결하여 가상 환경 속에서 재난 시나리오를 따라서 다양한 재난 상황을 경험하고, 상황에 알맞은 안전한 대처 방법을 학습한다. 또한, 감독관은 별도의 PC를 통해 체험자의 진행 상황을 실시간으로 모니터링하고, 필요시 즉각적인 피드백을 제공하여 교육 효과를 극대화한다.

- **재난 상황 시뮬레이션**
  - 다양한 재난 상황 시나리오를 구상하여 VR로 실감 나게 상황을 재현한다.
  - 시뮬레이션은 체험자가 실제 상황처럼 느낄 수 있도록 몰입감을 제공하며, 물리적 상호작용과 상황별 대처 방안을 연습할 수 있는 환경을 구축한다.
- **VR 기법 및 VR 기기 활용**
  - VR 기법을 사용하여 고품질의 시각적, 청각적 효과를 구현하여 현실감 있는 재난 체험을 제공한다.
  - VR 헤드셋, 컨트롤러와의 호환성을 보장하고, 사용자 UI가 직관적이며 사용자가 쉽게 조작할 수 있도록 설계해야 한다.
- **사용자 UI**
  - 체험자와 감독관이 시뮬레이션을 직관적으로 이해하고 사용할 수 있도록, 사

용자 UI는 직관적이고 간결하게 설계해야 한다. 체험자는 재난 상황에서 필요한 조작을 쉽게 수행할 수 있어야 하며, 감독관은 체험자의 진행 상태를 한눈에 파악할 수 있어야 한다.

- **1:1 서포트 및 피드백 시스템**
  - 시뮬레이션 중 체험자가 실시간으로 감독관의 지도를 받을 수 있는 1:1 서포트 기능을 제공해야 한다. 이 기능을 통해 감독관은 체험자의 행동을 모니터링하고 적절한 피드백을 즉각적으로 제공할 수 있다.
  - 감독관은 체험자 시점 화면을 모니터링하면서 동시에 탑 뷰로 전체 맵을 확인할 수 있어, 상황을 다각도로 파악하고 체험자에게 필요한 도움을 제공할 수 있다.
  - 체험자와 감독관 간의 원활한 의사소통을 위해 실시간 음성 통화 및 커맨드 입력 기능을 제공해야 하며, 이를 통해 즉각적이고 효율적인 상호작용이 이루어질 수 있도록 한다.
  - 모든 의사소통은 지연 없이 원활하게 이루어져야 하며, 시스템은 안정적인 연결 상태를 지속적으로 유지해야 한다.

## 2.2. 시스템 설계 상세화

### 2.2.1. 체험자

- **체험자 아바타**
  - 체험자의 시뮬레이션 경험의 몰입감을 높이기 위해 VR기기(Oculus)에서의 체험자 시선과 동일한 1인칭시점으로 제공한다.
  - Unity XR의 component로 XR Origin을 통하여 연동된 장비의 기준이 되는 오브젝트를 설정하고 VR 헤드셋의 기준 높이 등을 설정할 수 있다. XR Origin 내부 Left/Right Controller를 통해 양손 컨트롤러를 사용할 수 있다.
  - XR Ray Interactor 이용하면 컨트롤러가 향하는 직선 방향으로 레이저 포인터를 나타내어 체험자의 컨트롤러가 가리키는 부분을 정확히 보여준다.
  - XR Controller component를 통해 체험자는 Interactable 설정이 된 물체와 상호작

용을 할 수 있다.

- XR Interaction Toolkit이 제공하는 Locomotion System을 통해 체험자의 컨트롤러를 통해 이동 및 회전을 할 수 있다.

- **가상 환경 속 물체**
  - 현실에서의 물체 특성에 동일한 방식으로 여러 물체와 체험자가 상호작용할 수 있다.
  - 소화기: 한쪽 컨트롤러의 그립 버튼을 통해 소화기를 잡고 다른 한쪽 컨트롤러의 트리거 버튼을 통해 컨트롤러의 방향을 향해 소화 분말을 발사할 수 있다.
  - 벽돌/의자: 컨트롤러의 그립 버튼을 통해 벽돌이나 의자를 잡고 썸 스틱을 통해 회전시켜 창문을 깰 수 있다.

- **미니맵**
  - Unity의 Canvas를 사용하여 2D UI를 설계한다.
  - 단면도를 표시할 이미지를 UI Image 컴포넌트로 추가하고, 해당 단면도 이미지는 실제 VR 공간의 구조와 일치하게끔 제작한다.
  - 체험자의 위치를 실시간으로 반영하기 위해 체험자 아이콘을 단면도 위에 배치한다. 이 아이콘은 Unity의 RectTransform을 이용해 단면도의 좌표와 체험자의 3D 공간 좌표를 매핑하여 배치되게끔 구현한다.
  - 특정한 화면 크기에 맞춰서 확대 축소가 가능하도록 구현한다.

- **메뉴**

  **긴급 호출:**

  - 긴급 호출은 체험자가 시뮬레이션을 진행할 수 없는 긴급 상황인 경우, 감독관을 호출할 수 있는 기능이다.
  - 긴급 호출 시, 체험자와 감독관의 음성 채팅이 바로 시작될 수 있도록 구현한

다.

**인벤토리:**

- 인벤토리는 시뮬레이션 시작 시, 소지하게 된 물건들을 저장하여 체험자가 시뮬레이션에서 필요한 경우 사용할 수 있도록 도와주는 기능이다.
- 인벤토리의 물건을 선택하면 체험자의 손에 위치하여 특정 상호작용을 할 수 있다.

### 2.2.2. 감독관

#### 1) 단면도

**단면도 UI 구현**:

- Unity의 Canvas를 사용하여 2D UI를 설계합니다.
- 단면도를 표시할 이미지를 UI Image 컴포넌트로 추가하고, 해당 단면도 이미지는 실제 VR 공간의 구조와 일치하게끔 제작한다.
- 체험자의 위치를 실시간으로 반영하기 위해 체험자 아이콘을 단면도 위에 배치한다. 이 아이콘은 Unity의 RectTransform을 이용해 단면도의 좌표와 체험자의 3D 공간 좌표를 매핑하여 배치되게끔 구현한다.

**위치 동기화**:

- 체험자 프리팹의 Transform 컴포넌트를 이용해 position 값을 가져온다. 이 값은 3D 공간 내의 좌표로서, x, y, z 축에 대한 정보를 포함한다.
- 네트워크 전송은 Unity의 UNet 또는 Photon, NetCode 등을 활용해 구현한다. 체험자의 위치 데이터는 주기적으로 서버에 전송되며, 서버에서 이를 수신하여 단면도 상의 아이콘 위치를 갱신한다. (현재 Unet으로 진행 중이지만, 때에 따라 다른 네트워크 솔루션으로 마이그레이션 할 계획이다.)

#### 2) 컨트롤 메뉴

**안내 마커 생성**:

- 단면도에서 특정 위치를 클릭하면 해당 위치에 화살표 마커가 생성되도록 한다. 이 작업은 Unity의 Raycasting을 통해 사용자의 마우스 클릭 좌표를 감지하여 이루어진다.
- 클릭 된 좌표는 2D UI 상의 위치이므로, 이를 3D 가상 공간의 좌표로 변환해야 한다. 변환된 좌표에 마커를 생성하고, 마커는 체험자의 VR 환경에서도 동일한 위치에 표시되게끔 한다.
- ScreenPointToRay 메서드를 사용하여 2D UI 좌표를 기반으로 3D 레이(Ray)를 생성한다. 이 레이는 클릭 된 화면 위치에서 시작하여 3D 공간으로 쏘아진다.
- 레이가 3D 오브젝트의 Collider와 충돌하면, 충돌 지점의 hit.point가 클릭 된 3D 위치로 사용하게끔 한다.
- 마커는 프리팹(Prefab)으로 미리 준비해 두고, 클릭 시 인스턴스화하여 가상 공간에 배치하게끔 구현한다.

**위험 요소 생성**:

- 단면도의 특정 위치를 클릭했을 때, 위험 요소를 발생시키는 기능을 구현한다. 위험 요소는 낙석, 폭발 등 다양한 형태가 있을 수 있으며, 이를 선택할 수 있는 UI를 제공한다.
- 사용자가 위험 요소를 선택하면, 선택된 위험 요소가 발생할 위치를 클릭하여 해당 위치에 위험 요소가 나타나도록 한다. 위험 요소 역시 프리팹으로 구현하여, 클릭 된 위치에 인스턴스화한다.

**문 잠금 기능**:

- 단면도에 표시된 문을 클릭하면, 가상 공간에서 해당 문이 잠기는 기능을 구현한다. 문 오브젝트에는 문을 열고 닫는 기능을 제어하는 스크립트가 포함되어 있어야 하며, 클릭 이벤트 발생 시 해당 스크립트에서 잠금 상태를 변경할 수 있게끔 한다.

**음성 채팅 구현**:

- Unity에서 음성 채팅 기능을 구현하기 위해 Vivox 프로그램을 활용할 것이다.
- 음성 데이터는 네트워크를 통해 실시간으로 전송되며, 감독관과 체험자는 서로의 음성을 들을 수 있게끔 구현한다.
- UI 상에 음성 채팅을 on/off 할 수 있는 버튼을 제공하고, 이 버튼을 통해 마이크 입력을 제어할 수 있으며 음성 채팅 기능은 네트워크 성능과 음질을 고려하여 최적화시킨다.

### 3) 시스템 메뉴

**카메라 전환**:

- 체험자 프리팹에 Camera 컴포넌트를 추가한다. 이 카메라는 체험자의 시점을 보여주는 역할을 한다.
- 카메라의 Tag를 MainCamera로 설정하여, Unity에서 기본 카메라로 인식하게 한다. 감독관의 화면에서 체험자의 카메라 시점을 전환할 수 있도록 스크립트를 작성한다. 체험자의 카메라를 활성화하여 단면도 대신에 비추게끔 구현한다.
- 전환 시 매끄러운 화면 전환을 위해 페이드 인/아웃 효과를 적용한다.

**시뮬레이션 초기화**:

- 시뮬레이션 도중 체험자의 위치나 상태를 초기화할 수 있는 기능을 제공한다. 이 기능은 단순히 체험자의 위치를 초기 위치로 이동시키는 것뿐만 아니라, 시뮬레이션 내 모든 오브젝트의 상태도 초기 상태로 되돌려야 한다.
- 초기화 기능은 전체 시뮬레이션의 상태를 관리하는 관리 스크립트를 통해 구현하며, 초기화가 실행되면 해당 스크립트가 각 오브젝트에 초기 상태를 명령하는 방식으로 구현한다.

**일시정지**:

- 일시정지는 시뮬레이션의 모든 동작을 일시정지시키는 기능이다. Unity에서는 Time.timeScale을 0으로 설정하여 게임 내 모든 물리적 움직임과 애니메이션을 멈추게끔 구현한다.
- 일시정지 상태에서도 사용자 인터페이스(UI)는 동작하도록 설계하여, 감독관이 추가적인 조치를 취할 수 있게 한다.

**시뮬레이션 종료**:

- 시뮬레이션 종료는 네트워크 연결을 해제하고, 체험자와 감독관의 세션을 종료하는 기능으로, 종료 시, 네트워크 연결 해제를 처리하고 시뮬레이션을 안전하게 종료할 수 있도록 해야 한다.
- 종료 시에는 체험자에게도 종료 알림을 제공하여, 종료 시점을 인지할 수 있게 한다.

### 2.2.3. 재난 상황 환경 구현

#### 1) 감독관의 역할

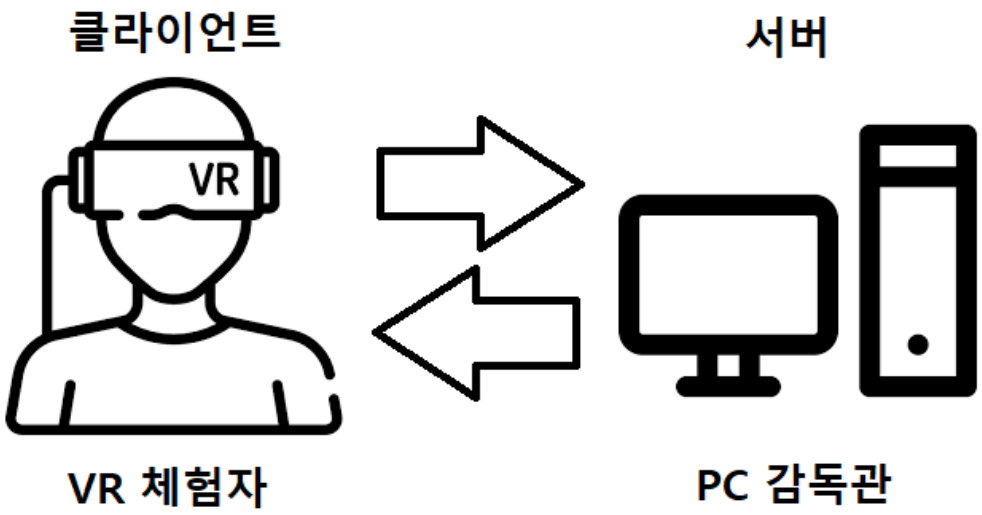
- **설정 기능**: 네트워크 연결이 완료되면, 감독관은 PC를 통해 체험자의 위치, 화재 발생 위치, 체험자가 소지한 물품 등을 설정할 수 있다.
  - **위치 설정**: 유니티의 dropout 기능을 이용해 체험자와 화재의 위치를 1, 2, 3층 중에 정할 수 있다. 이를 통해 다양한 재난 상황을 설정할 수 있다.
  - **물품 설정**: 체크박스를 통해 체험자가 소지하고 사용할 수 있는 물품(예: 스마트폰, 손수건)을 선택하여 시나리오에 맞는 물품이 제공된다.
- **시나리오 구성**
  - 감독관은 체험자의 위치를 1층에, 화재를 위층(2층 또는 3층)에 배치하여 체험자가 단순히 탈출하는 간단한 상황을 설정 가능하다.
  - 화재를 체험자보다 아래층에 배치하고, 소지 물품으로 소화기나 스마트폰을 제공하여, 체험자가 화재를 진압하거나 구조 요청 등의 복잡한 대처를 해야 하는 시나리오도 설정 가능하다.

- **모니터링과 돌발 상황**
  - 설정이 완료 후, 감독관은 카메라를 통해 체험자의 행동을 실시간으로 모니터링할 수 있으며, 낙석, 폭발과 같은 돌발 상황을 발생시킬 수 있다.
  - 체험자가 잘못된 행동을 할 경우, 일시정지 기능을 통해 시뮬레이션을 멈추고 마이크로 올바른 대처 방법을 즉각적으로 지시할 수 있다.

### 2) 체험자의 역할

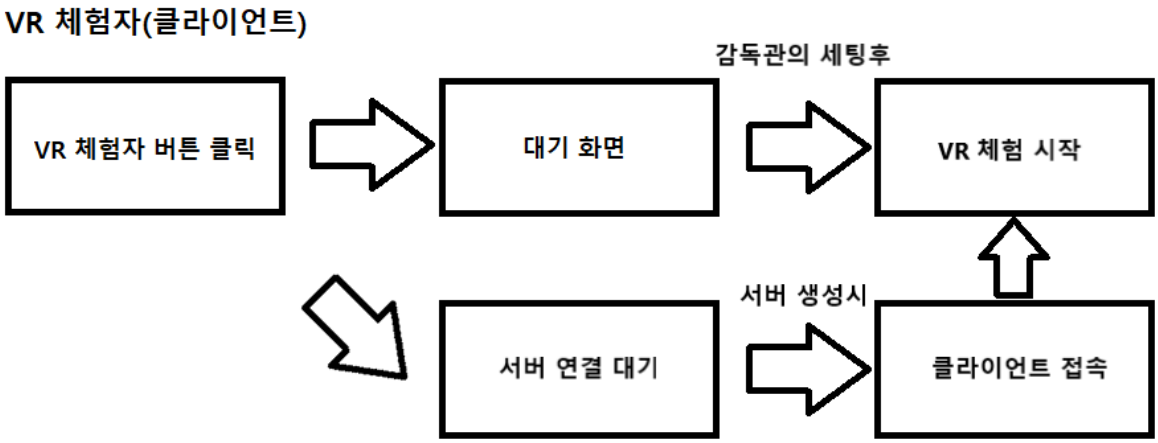
- **VR 환경 내에서의 시작**: 체험자는 화재가 발생한 가상의 건물 내부에서 시작하며, 주어진 재난 상황에 맞춰 빠르게 대처해야 한다. 감독관이 설정한 상황에서 주어진 물품을 활용해 빠르게 대처해야 하며, 잘못된 대처를 할 경우 감독관의 피드백을 통해 올바른 대처 방법을 학습하게 된다.
- **물품 활용 및 대처**: 체험자는 감독관이 설정한 물품을 활용하여 상황에 적절히 대처해야 하며, 잘못된 행동을 할 경우 감독관의 피드백을 통해 올바른 방법을 학습한다.

## 2.3. 시스템 구성도



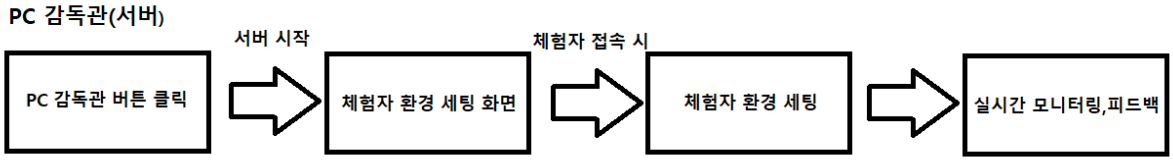

[ 그림 1 ] 기본 동작

시스템은 기본적으로 서버 역할을 하는 PC 감독관과 클라이언트 역할의 VR 체험자로 이루어져 있다.




[ 그림 2 ] VR체험자(클라이언트)의 흐름

먼저 VR 체험자의 경우 VR 장비를 통해 접속한다. 이후 첫 UI를 통해 VR 체험자 버튼을 선택하게 되면 대기화면으로 들어감과 동시에 서버의 연결을 기다린다. 이후 감독관이 접속하여 서버를 생성하면 클라이언트로 접속하게 되고 감독관이 체험자의 환경 세팅을 끝내고 나면 VR 체험이 시작된다.




[ 그림 3 ] PC 감독관(서버)의 흐름

PC 감독관의 경우는 개인 Laptop을 통해 접속한다. 이후 첫 UI에서 PC 감독관 버튼을 선택하게 되면 서버가 시작되고 체험자 환경 세팅 화면으로 진입한다. 세팅 화면에서는 체험자가 접속하면 세팅이 동작하며 세팅이 끝나고 시작 버튼을 누르면 VR 체험자도 체험을 시작하고, 실시간 모니터링을 시작한다.

## 2.4. 개발 환경

- **개발 도구**: Unity 2022.3.35f1, Netcode for GameObjects, XRInteraction Toolkit

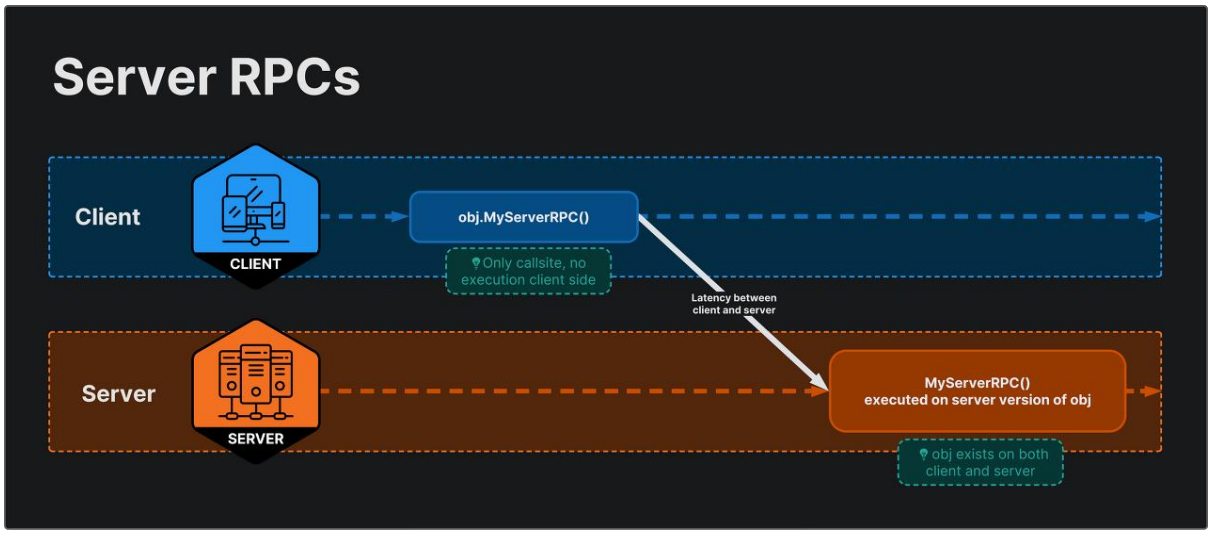
- **프로그램 언어**: C#
- **하드웨어**: Laptop, Oculus Quest2
- **버전관리**: Unity Version Control

# 3. 과제 구현 내용

## 3.1. 네트워크 환경 구성

본 과제에서는 VR 기반 재난 시뮬레이션 시스템을 구현하기 위해 Unity의 네트워크 솔루션인 **Netcode for GameObjects**를 사용했다. 이 시스템은 감독관과 체험자 간의 원활한 실시간 상호작용을 지원하도록 설계됐다.

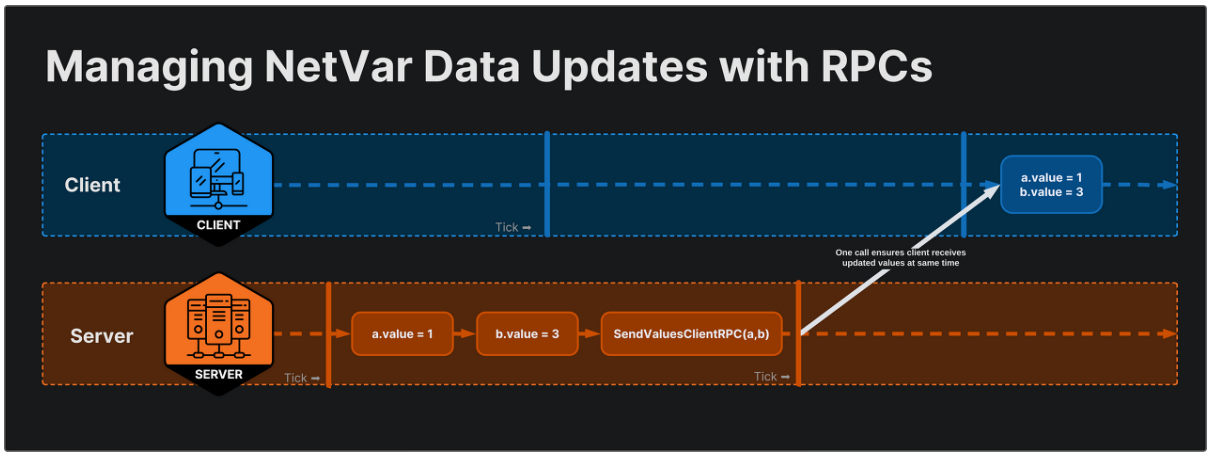
Netcode는 유니티가 지원하는 네트워크 라이브러리 중 하나로, Server Authoritative의 특징을 가지고 있다. 클라이언트에서 발생한 행동은 서버에서 처리되며, 서버가 게임 상태를 관리하고 클라이언트가 서버에 행동을 요청하면, 서버가 해당 요청을 검증하고 게임 상태를 동기화해 업데이트한다. 이 동기화를 위해 Netcode는 원격 프로시저 호출인 **RPC(Remote Procedure Call)**를 이용하는데, RPC는 클라이언트와 서버 간의 메시지를 전송하는 방식이다.




[ 그림 4 ] Server RPC 동작

RPC는 ServerRpc와 ClientRpc로 나뉘며, **ServerRpc**는 클래스 내에서 함수를 호출하는 것

과 같고, 클라이언트에 의해서만 호출해 서버로 정보를 보낼 수 있으며, 항상 서버에서 수신되고 실행된다.




[ 그림 5 ] Client RPC 동작

**ClientRpc**의 경우, 서버가 ClientRpc를 호출하여 모든 클라이언트에서 함수를 실행하도록 할 수 있고, 특정 클라이언트의 ID를 안다면 그 클라이언트에서만 함수를 실행하는 것도 가능하다.

이번 과제에서는 먼저 감독관이 호스트로서 서버를 열고, 체험자가 클라이언트로 접속해 자신의 플레이어 프리팹을 스폰하기 위해 코드상에서 ServerRpc를 이용해 서버에 요청한다.

```
[ServerRpc(RequireOwnership = false)]
1 reference
private void RequestSpawnXROnServerServerRpc(ulong clientId, ServerRpcParams rpcParams = default)
{
    Debug.Log("VRPlayer spawned on server.");
    GameObject xrOriginPrefab = Instantiate(XROrigin);
    NetworkObject netObj = xrOriginPrefab.GetComponent<NetworkObject>();
    xrOriginPrefab.SetActive(true);
    netObj.SpawnAsPlayerObject(clientId);
    //xrOriginPrefab.GetComponent<NetworkObject>().SpawnWithOwnership(clientId);
    StartCanvas.SetActive(false);
    playerCamera = xrOriginPrefab.GetComponentInChildren<Camera>();
    playerCamera.targetDisplay = 0;
}
```

[ 그림 6 ] ServerRpc를 통한 체험자 플레이어 프리팹 스폰 요청 함수

- **RequestSpawnXROnServerServerRpc**: 클라이언트가 호출할 수 있는 서버 측 메서드이다. ServerRpc를 통해 클라이언트가 서버에 VR 플레이어 오브젝트를 요청하고, 이 요청을 통해 서버에서 VR 플레이어가 스폰된다.

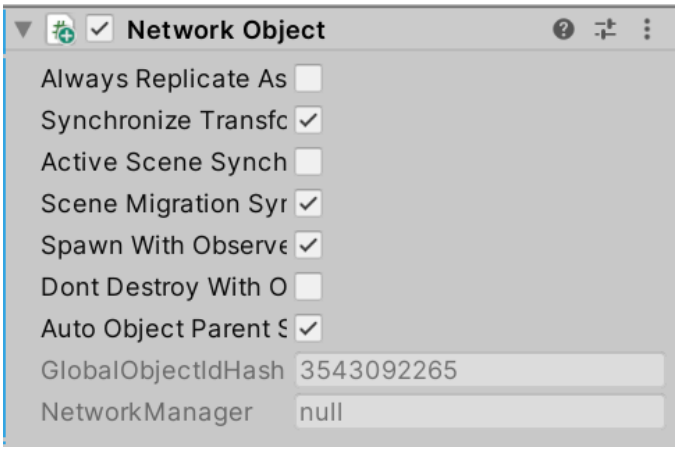
- **SpawnAsPlayerObject(clientId)**: 이 메서드는 서버에서 플레이어 오브젝트를 해당 클라이언트의 소유권을 부여한 상태로 스폰한다.
- **RequireOwnership = false**: 이 옵션을 통해 소유권이 없는 클라이언트도 이 RPC를 호출할 수 있도록 한다. 이는 클라이언트가 자신의 플레이어 오브젝트를 서버에 요청할 수 있게 만든다.

Unity Netcode에서는 네트워크에서 동기화해야 하는 오브젝트는 **NetworkObject**를 통해 처리된다. 즉, 네트워크에서 상호작용하는 모든 오브젝트는 NetworkObject component를 가져야 하고, 이를 통해 서버와 클라이언트 간의 동기화가 이루어진다.

```
GameObject xrOriginPrefab = Instantiate(XROrigin);
NetworkObject netObj = xrOriginPrefab.GetComponent<NetworkObject>();
xrOriginPrefab.SetActive(true);
netObj.SpawnAsPlayerObject(clientId);
```

[ 그림 7 ] 플레이어 프리팹을 네트워크 오브젝트로 추가하는 코드

- **NetworkObject**: 네트워크상에서 오브젝트를 제어하고 동기화하는 기본 component다. 이 component를 가진 오브젝트만이 네트워크상에서 여러 클라이언트와 서버 간에 동기화될 수 있다.
- **SpawnAsPlayerObject**: 네트워크 오브젝트를 클라이언트의 플레이어 오브젝트로 스폰한다. 여기서 clientId는 소유권을 부여받을 클라이언트의 ID를 나타낸다.

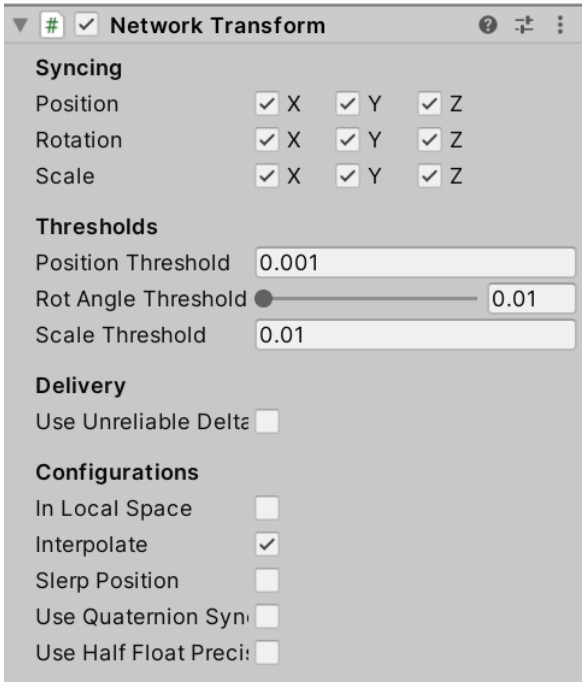



[ 그림 8 ] 네트워크 오브젝트 컴포넌트 인스펙터

[그림 8]은 어떤 게임 오브젝트에 네트워크 오브젝트 component를 추가한 것이다.

네트워크에서 플레이어가 어떤 상태에서 다른 상태로 이동하는 과정을 관리하는 것이

**Network Transition**이다. 예를 들어, 게임을 시작할 때 감독관과 체험자가 각각 자신에게 맞는 scene으로 로드되며, 각 scene에서 플레이어 오브젝트가 생성되어 네트워크에서 동기화된다.




[ 그림 9 ] 네트워크 트랜스폼 컴포넌트 인스펙터

위는 게임 오브젝트에 네트워크 트랜스폼 component를 추가한 것이다.
이를 바탕으로 감독관과 체험자 사이의 통신을 구현하고 네트워크상의 제어가 가능해졌다.

## 3.2. 시뮬레이션 시나리오 구현

### 3.2.1. VR체험환경(3D) 구성


[ 그림 10 ] 사용 3D 건물

VR 체험자의 체험 환경은 기본적으로 3층 건물 내부에서 감독관이 세팅 UI에서 정한 층수의 방안에서 시작된다.


[ 그림 11 ] 방 내부구조

방의 전체적인 구성은 [그림 11]과 같다.

[ 그림 12 ] 구비 물품

VR 체험자는 감독관이 세팅한 환경에 따라 [그림 12]와 같은 방구석의 소화기나 책상 위의 스마트폰, 손수건 등을 이용하여 상황에 맞는 재난 상황 시뮬레이션을 체험한다.

### 3.2.2. Ui구성

#### 3.2.2.1. 시작UI


[ 그림 13 ] 시작 UI

시작 Ui는 [그림 13]와 같으며, VR 체험자 버튼을 누르면

```
public void StartClient()
{
    NetworkManager.Singleton.StartClient();
    StartCoroutine(WaitForClientConnection());

    StartCanvas.SetActive(false);
    WaitingCanvas.SetActive(true);
    //playerCamera.targetDisplay = 0;
}
```

[ 그림 14 ] vr체험자 버튼에 적용된 StartClient()함수

**StartClient()** 함수가 실행되고 앞서 적용한 Netcode를 이용하여 VR 체험자는 클라이언트로 접속하며, **SetActive**를 통해 대기화면으로 넘어간다.

```
private IEnumerator WaitForClientConnection()
{
    while (!NetworkManager.Singleton.IsClient || !NetworkManager.Singleton.IsConnectedClient)
    {
        Debug.Log("Waiting for connection...");
        yield return new WaitForSeconds(1f);
    }

    // 연결 후 VR 씬 로드
    LoadVRSceneServerRPC(NetworkManager.Singleton.LocalClientId);
}
```

[ 그림 15 ] WaitForClientConnection()함수

```
[ServerRpc(RequireOwnership = false)]
참조 1개
public void LoadVRSceneServerRPC(ulong clientId)
{
    if (!NetworkManager.Singleton.IsServer) return; // 서버에서만 실행

    // VR 플레이어를 생성하고 네트워크 객체로 스폰
    NetworkPlayer = Instantiate(VRPlayer);
    NetworkObject netObj = NetworkPlayer.GetComponent<NetworkObject>();
    netObj.SpawnAsPlayerObject(clientId, true); // 클라이언트 ID에 맞는 네트워크 객체 스폰
    NetworkObjectManager.Instance.RegisterNetworkObjectServerRpc(netObj.NetworkObjectId);
}
```

[ 그림 16 ] LoadVRSceneServerRPC()함수

여기서 클라이언트는 **WaitForClientConnection()**을 통해 서버의 연결을 기다리는데, 서버가 연결되면 **LoadVRSceneServerRPC()** 함수를 실행하여 체험자에 맞는 네트워크 오브젝트를 생성해주고, 체험자의 네트워크 오브젝트를 이용하여 위치를 변경하거나 관찰하는 카메라를 만들기 위해 **NetworkObjectManager**에 저장해둔다.

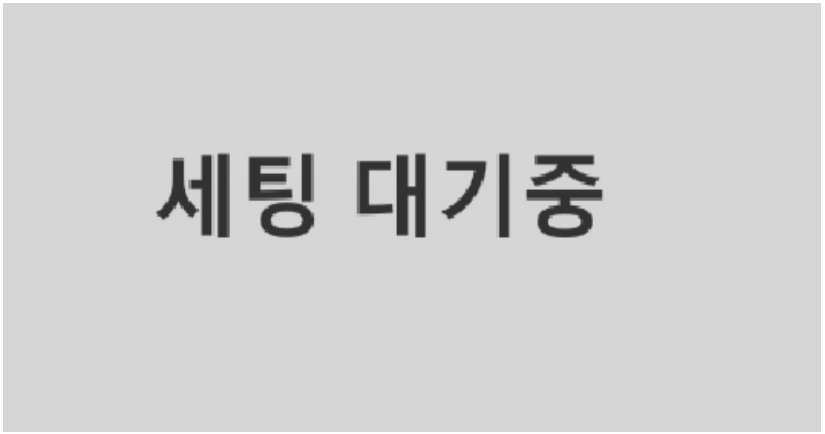
다음으로 PC감독관 버튼을 누르면

```
public void StartHost()
{
    NetworkManager.Singleton.StartHost();
    LoadPCSceneServerRPC(NetworkManager.Singleton.LocalClientId); // Host는 PC 씬을 로드
}
public void LoadPCSceneServerRPC(ulong clientId)
{
    if (!NetworkManager.Singleton.IsServer) return; // 서버에서만 실행

    // PC 설정 캔버스 활성화 및 StartCanvas 비활성화
    StartCanvas.SetActive(false);
    SettingCanvas.SetActive(true);

}
```

[ 그림 17 ] pc 감독관 버튼에 적용된 StartHost()함수

**StartHost()** 함수가 실행되어 PC 감독관은 서버의 호스트로 접속하며, 세팅 화면으로 넘어간다.

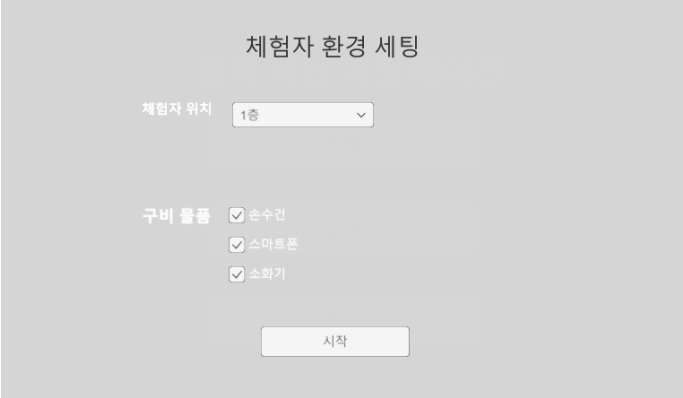
### 3.2.2.2. 대기UI



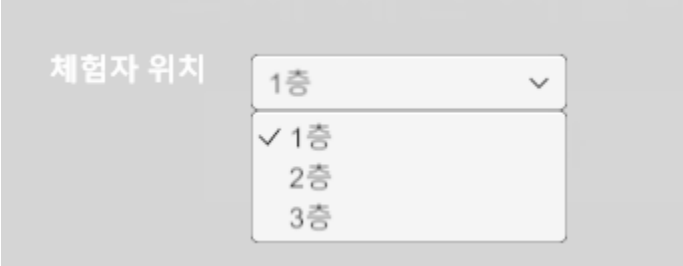

[ 그림 1 ] 대기UI

체험자는 VR 체험자 버튼을 누른 후 위 사진과 같이 대기 화면으로 진입하고 감독관의 환경 세팅이 끝나게 되면 건물 내부에서 시작하게 된다.

### 3.2.2.3. 세팅UI




[ 그림 19 ] 세팅 UI




[ 그림 20 ] 체험자 위치를 조정하는 Dropdown

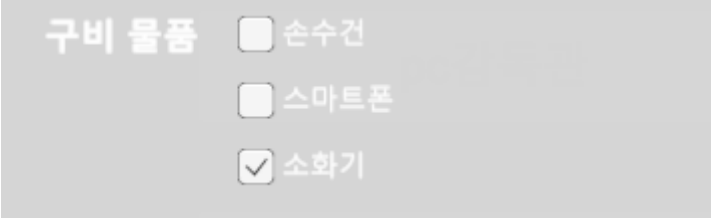
감독관은 PC 감독관 버튼을 누르면 위와 같이 세팅 화면으로 진입하고, 원하는 위치를 위 사진과 같은 Dropdown을 통해 지정해준다.

```
switch (index)
{
    case 0:
        TeleportToPosition(firstFloor);
        Debug.Log("Player Location Change(1st)");
        break;
    case 1:
        TeleportToPosition(secondFloor);
        Debug.Log("Player Location Change(2nd)");
        break;
    case 2:
        TeleportToPosition(thirdFloor);
        Debug.Log("Player Location Change(3rd)");
        break;
}
```

```
private void TeleportToPosition(Vector3 newPosition)
{
    if (IsServer)
    {
        netObj.transform.position = newPosition;
        RequestPositionChangeClientRpc(newPosition);

        Debug.Log("Request Teleport");
    }
```

[ 그림 21 ] Dropdown에 적용된 코드

[그림 21]의 Switch문을 이용하여 층수에 맞는 방 위치로 이동시킨다. 여기서 netObj는 시작 UI에서 저장한 체험자의 NetworkObject이며, 위와 같은 방식으로 ServerRpc, ClientRpc를 이용하여 netObj의 transform을 이동시켜 바뀐 netObj의 위치를 동기화시킨다.




[ 그림 22 ] 구비 물품을 설정하는 Toggle

```
private void OnIsActiveChanged(bool previousValue, bool newValue)
{
    foreach (var obj in networkObjects)
    {
        obj.gameObject.SetActive(newValue);
        Debug.Log($"Setting {obj.name} active state to {newValue}");
    }
}
```

[ 그림 23 ] Toggle에 적용된 코드

또한 감독관은 [그림 23]과같이 물품의 체크박스를 이용하여 체험자가 이용할 수 있는 물건을 체크박스의 값(눌림 값)에 따라 세팅할 수 있다.

```
public void ShowManagerCanvas()
{
    if (NetworkObjectManager.Instance.GetNetworkObject() != null)
    {
        SettingCanvas.SetActive(false);
        VrPlayerViewCamera.SetActive(true);
        ManagerCanvas.SetActive(true);
        InformPlayerLocation.SetActive(true);
        CameraSettingClientRpc();
    }
```

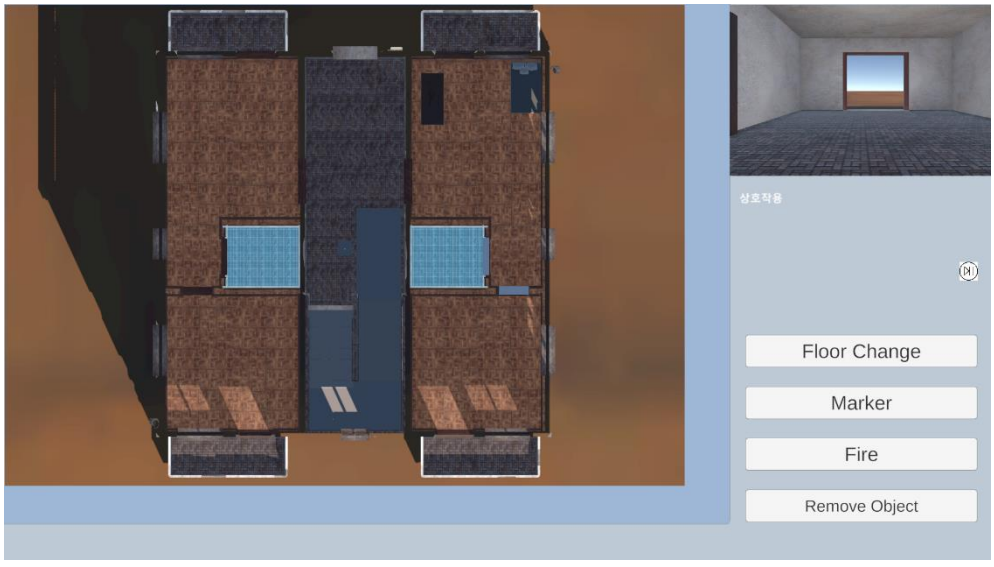
[ 그림 24 ] ShowmanagerCanvas()함수

```
[ClientRpc]
참조 1개
private void CameraSettingClientRpc()
{
    if (NetworkManager.Singleton.IsHost)
    {
        // Host의 카메라는 따로 설정하지 않음
        return;
    }
    WaitingCanvas.SetActive(false);
    playerCamera.targetDisplay = 0;
}
```

[ 그림 25 ] CameraSeetingClientRpc()함수

이후 모든 세팅이 끝나고 감독관이 시작 버튼을 누르면 **ShowManagerCanvas()** 함수가 실행되고 관리화면으로 넘어가면서 **CameraSettingClientRpc()** 함수를 이용하여 체험자를 대기화면에서 체험 환경으로 진입시킨다.

### 3.2.2.4. 관리 UI




[ 그림 26 ] 관리 UI

감독관이 세팅을 마치고 관리 화면으로 넘어가게 되면 [그림 26]의 좌측에 있는 미니맵과 우측상단에 있 체험자의 작은 화면을 통해 실시간 모니터링이 가능하고 버튼을 이용하여 체험자와 감독관 간의 의사소통이 가능하다.

### 3.2.3. 체험자와 감독관 간의 의사소통 기법

#### 3.2.3.1. 안내 마커와 오브젝트 스폰


[ 그림 27 ] 감독관 화면에서 안내 마커를 스폰했을 때

[ 그림 28 ] 체험자 화면에서 안내 마커가 스폰됐을 때

[그림27]의 감독관 측의 화면에 나타난 건물의 단면도에서 어느 위치를 클릭하게 되면 해당 위치에 체험자를 안내하는 역할을 하는 '마커'가 스폰된다. 마커는 최대 1개 스폰할 수 있고, 마커가 이미 스폰 되어지고 있는 동안에 감독관이 다른 위치를 클릭하면 이미 스폰된 마커는 사라지고 새로운 위치에 마커가 스폰된다.

이 마커를 통해 시뮬레이션에 기대할 수 있는 이점은 다음과 같다.

- **실시간 안내**: 감독관이 클릭하는 즉시 마커가 생성되므로, 체험자는 신속하게 안내받을 수 있다. 체험자가 특정 위치로 이동하거나 행동을 취해야 하는 상황에서 유용하다.
- **명확한 시각적 피드백**: 마커는 시각적 피드백을 제공해 체험자가 현재 목표 위치를 쉽게 인식할 수 있도록 돕는다. 이는 특히 복잡한 건물 내부에서 길을 잃을 수 있는 상황에서 중요한 역할을 한다.
- **유연한 위치 조정**: 이미 스폰된 마커가 새로운 클릭으로 대체되기 때문에, 실시간으로 체험자의 경로를 변경하거나 상황에 따라 새로운 위치로 안내할 수 있는 유연성을 제공한다.
- **오류 방지**: 동시에 하나의 마커만 스폰되기 때문에, 여러 마커로 인해 체험자가 혼란을 겪거나 잘못된 위치로 갈 위험이 줄어든다.
- **감독관의 능동적 개입**: 감독관이 직접 위치를 선택함으로써, 시뮬레이션 과정에서 체험자에게 중요한 정보나 지시를 전달하는 능동적인 피드백이 가능하다.


[ 그림 29 ] 감독관 화면에서 불을 스폰했을 때

[그림 29]에서 우측 하단의 Fire 버튼을 클릭하면 감독관이 단면도를 클릭 시, 스폰되는 오브젝트를 마커에서 불 오브젝트로 변경할 수 있다. 불은 기본적으로 최대 5개(에디터에서 임의로 설정 가능)까지 스폰할 수 있으며, 건물에 최대 개수의 불이 스폰되어져 있는 상태에서 추가로 스폰하면 가장 먼저 스폰된 불이 사라지고 해당 위치에 새로운 불이 생성되게끔 구현하였다.

감독관이 임의로 화재 상황을 재현할 수 있을 때 시뮬레이션에 기대할 수 있는 이점은 다음과 같다.

- **동적 시나리오 생성**: 감독관이 전략적이거나 어려운 위치에 불을 배치함으로써 시나리오를 각 세션마다 유연하게 조정하고 독특하게 만들수 있다.

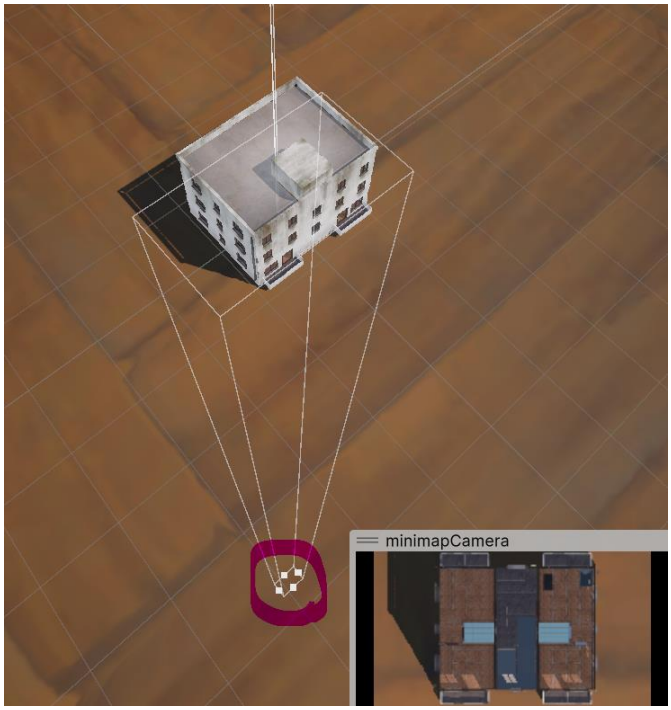
- **실시간 난이도 조절**: 체험자의 성과에 따라 불을 추가하거나 제거함으로써 난이도를 실시간으로 조절할 수 있다.

- **참가자의 집중 유도**: 불의 위치를 제어함으로써 체험자의 주의를 중요한 영역으로 유도하고, 시뮬레이션의 현실감과 긴박감을 높일 수 있다.

- **특정 기술 훈련**: 다양한 위치에 화재를 배치함으로써 긴급 상황에서의 이동 및 의사 결정 능력을 테스트할 수 있다.

- **화재의 확산 시뮬레이션**: 감독관이 실시간으로 화재가 확산하는 상황을 시뮬레이션하여 복잡한 환경에서 재난이 어떻게 진행되는지 더 몰입감 있게 경험할 수 있다.

스폰된 마커와 불은 우측 하단의 **Remove Object** 버튼을 클릭하면 가상환경상에서 전부 제거된다. 위 요소들을 통해 몰입감 높은 훈련 환경을 제공할 수 있으며, 실시간 피드백이 가능하고 다양한 시나리오를 시뮬레이션할 수 있다.
단면도를 클릭하면 해당 위치의 포지션 정보를 알아내고 오브젝트를 스폰하는 원리는 다음과 같다.




[ 그림 30 ] 건물의 단면도를 비추기 위한 카메라

[그림 30]을 보면 아래에서 건물을 수직으로 비추는 카메라가 하나 있다. 해당 카메라에는 건물의 단면도가 비치고 이 단면도의 어느 위치를 감독관이 클릭 시, 클릭한 위치를 기준으로 카메라에서 레이를 쏘고, 해당 레이가 충돌한 지점(Collider)을 월드 좌표로 변환한다. 이 좌표(포지션)에 오브젝트를 스폰시키는 원리이다.

```
private Vector3 MiniMapToWorldPosition(Vector2 normalizedPoint)
{
    // 미니맵 카메라에서 레이 생성
    Ray ray = miniMapCamera.ViewportPointToRay(new Vector3(normalizedPoint.x, normalizedPoint.y, miniMapCamera.nearClipPlane));

    // 레이캐스트를 위한 레이캐스트 히트 정보 생성
    RaycastHit hit;

    // BoxCollider가 있는 오브젝트와 충돌 체크
    if (Physics.Raycast(ray, out hit))
    {
        // BoxCollider와 교차점(hit.point)을 반환
        return hit.point;
    }

    // 교차점이 없으면 기본값 반환
    return Vector3.zero;
}
```

[ 그림 31 ] 레이캐스트를 이용해 교차점을 반환하는 코드

### 3.2.3.2. 체험자 시점을 실시간 모니터링


[ 그림 32 ] 우측 상단에 체험자 시점을 제공하는 화면

감독관은 자신의 화면의 우측 상단을 통해 체험자의 시점을 실시간으로 확인할 수 있다. **NetworkObjectManager**에 저장한 체험자의 **NetworkObject**를 카메라가 따라가게 만들고, **RawImage**를 이용해 실시간으로 시점을 반영할 수 있게 구현됐다.

```
private void LateUpdate()
{
    if (target != null)
    {
        transform.position = target.position;
        transform.rotation = target.rotation;
    }
}
```

[ 그림 33 ] 카메라가 체험자를 따라가기 위해 작성된 코드

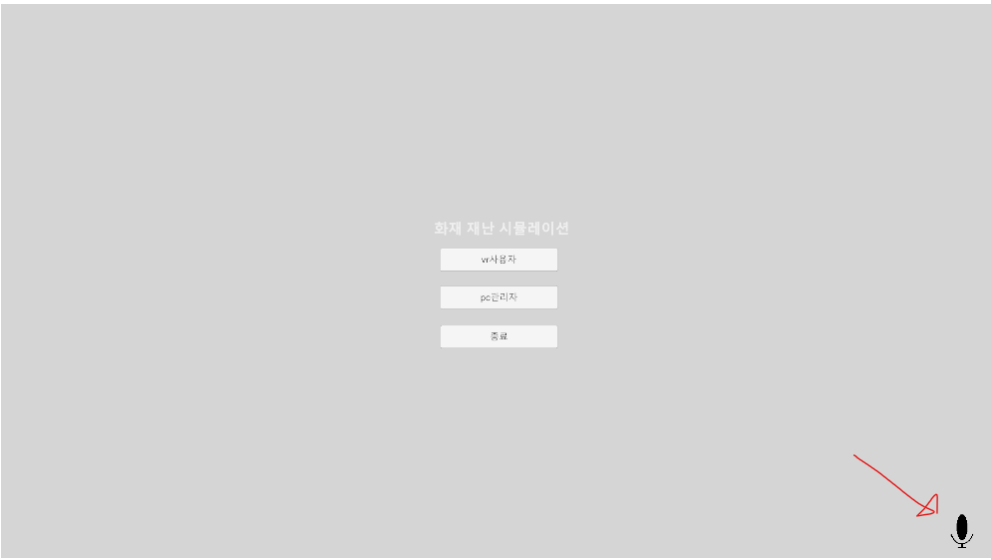
### 3.2.3.3. Vivox를 이용한 음성 채팅

패키지 매니저를 통해 프로젝트에 Vivox를 추가하였다.

Vivox는 멀티플레이어 커뮤니케이션을 위한 Unity의 음성 및 문자 채팅 서비스로 관리형 호스팅 솔루션을 통해 음성 채팅 및 다이렉트 메시지 문자 서비스를 제공한다. 게임에 연결하고 프로젝트 설정을 지정하여 즉시 프로젝트에 커뮤니케이션을 추가할 수 있다.

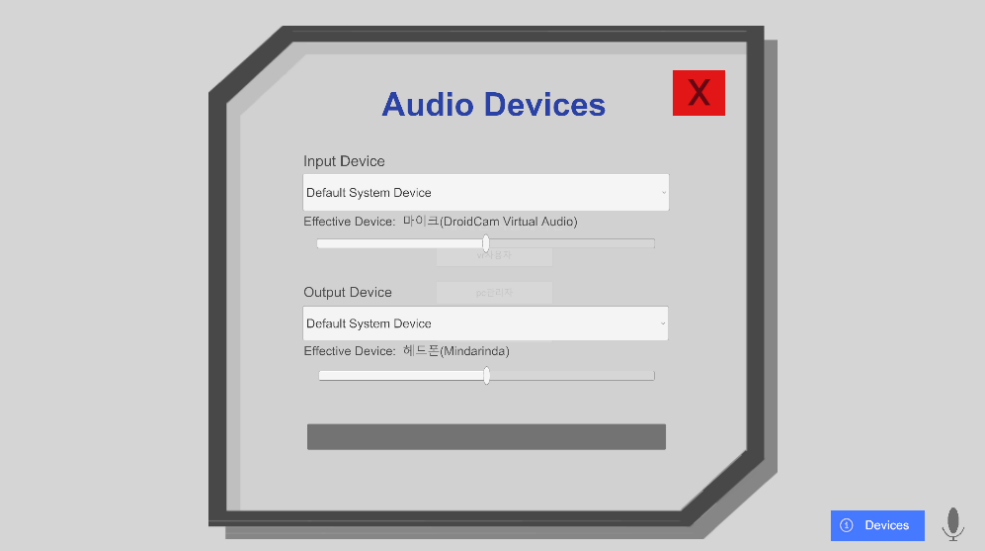
2D 및 3D 채널에서 사용자 수에 제한 없이 연결할 수 있으며, 대시보드에서 동시 실행을 모니터링할 수 있다. 커스텀 Ui를 추가한다면 플레이어가 음성 볼륨을 제어하고, 음소거 하고, 채널을 관리할 수 있게 해준다.

Vivox에서 제공하는 기본적인 UI와 샘플 코드를 최대한 활용하여 보이스 챗에 접속하고 입출력 디바이스 설정 및 소리 크기 조절 기능을 추가하였다.




[ 그림 2 ] 시작 화면에서 우측 하단에 배치된 보이스챗 로그인 아이콘

[그림 34]와 같이 프로그램 실행 직후 나오는 초기 화면에서 하단 우측에 자리잡은 마이크 아이콘을 클릭하고 잠시 기다리면 그 옆에 푸른색의 Devices 버튼이 생성되고 Vivox 보이스챗 서버에 로그인하게 된다. (이때, 감독관과 체험자 양쪽에서 로그인을 완료해야 한다.)



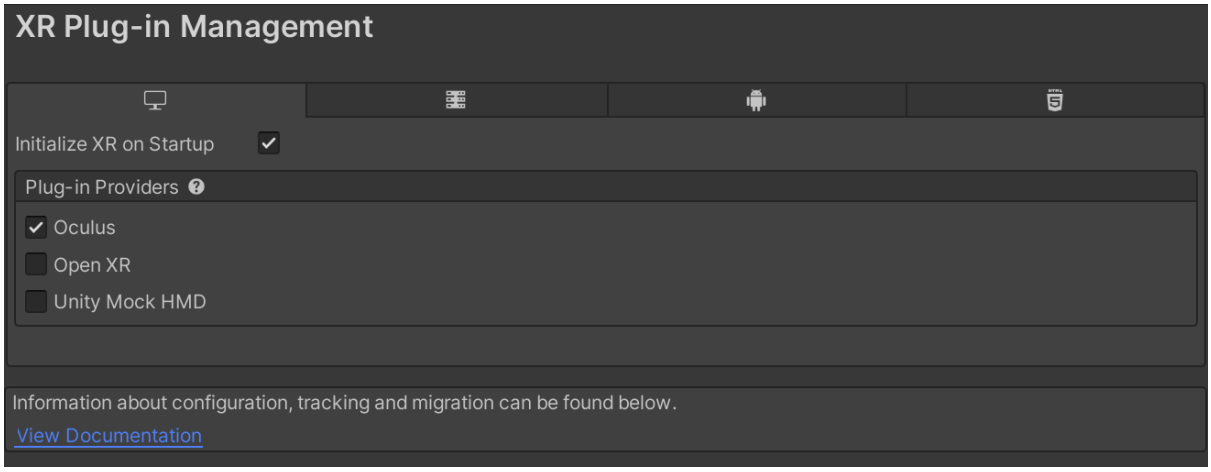

[ 그림 3 ] 오디오 입출력 디바이스 설정창

Devices 버튼을 클릭하여 나오는 메뉴를 통해 입출력 디바이스를 지정할 수 있는 또한, 소리 크기를 조절할 수 있다. 하단의 짙은 회색 바를 통해 마이크 입력이 잘 이루어지는지 확인할 수 있다.

이렇게 세팅이 끝나면 시뮬레이션 중에 감독관과 체험자 간에 대화를 통한 의사소통이 가능해진다.

## 3.3. VR 환경 구현

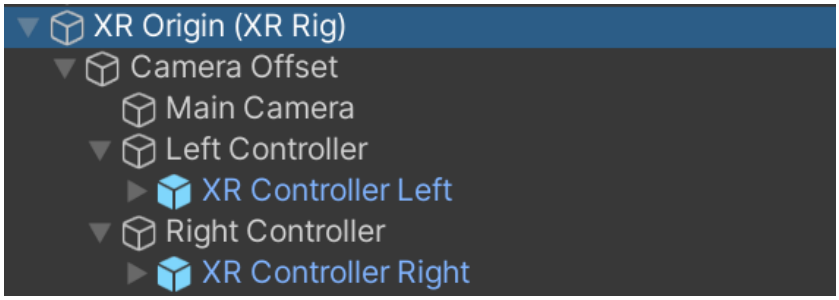
### 3.3.1. XR Origin




[ 그림 36 ]

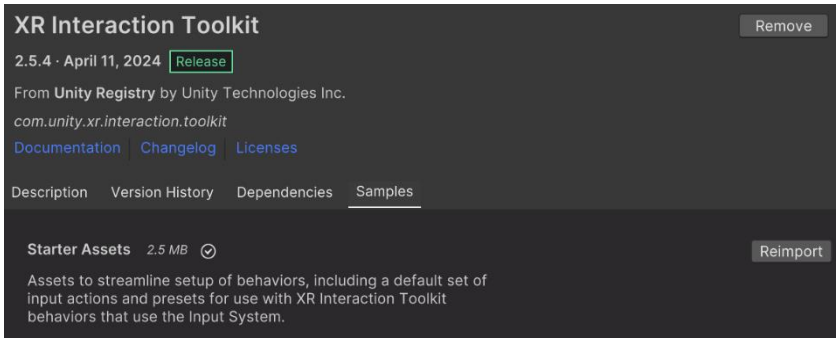
### 3.3.1.1. XR Plug-in Management

유니티에서 지원하는 VR 장비들을 모바일과 PC에 연동하기 위해 XR Plug-in Mnagement Package를 설치한 이후 설정값을 Oculus로 선택한다. 본 과제에서 사용하는 VR 장비인 Oculus의 컨트롤러의 위치와 방향을 연동할 수 있게 해주며, Unity UI와의 상호작용 등을 component만으로도 사용할 수 있게 해준다.




[ 그림 37 ] XR Origin 오브젝트

유니티에서 연동된 VR 장비의 기준이 되는 오브젝트를 설정하기 위해 XR Origin을 추가한다. 기본적으로 카메라와 컨트롤러가 설정되어 있다. 여기서 컨트롤러가 화면이 표시될 수 있도록 컨트롤러 프리팹을 왼쪽, 오른쪽 컨트롤러에 각각 추가한다. 다음은 XR Origin 오브젝트에 추가된 여러 component에 대한 설명이다.




[ 그림 38 ] XR Interaction Toolkit Package

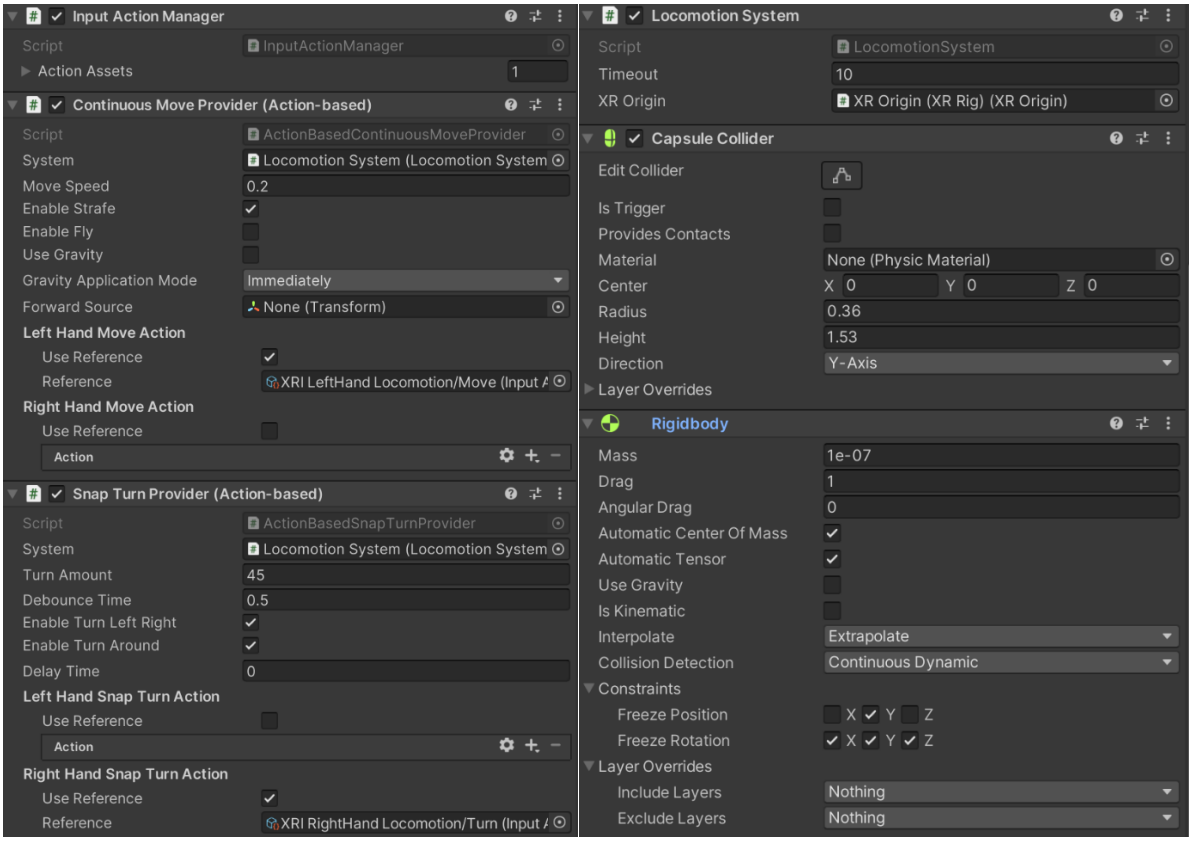
### 3.3.1.2. XR Interaction Toolkit

VR 장비의 입력을 받아 상호작용이 가능하게 하기 위해 XR Interaction Toolkit Package를 설치해 준다.

XR Interaction ToolKit이란 Unity에서 제작한 Unity XR 기반의 플러그인이다. VR의 일반

적인 기능을 스크립트 없이 편하게 구현하게 해주며, 유니티에서 지원하는 다양한 VR 장비들을 모바일과 PC에 손쉽게 연동할 수 있게 해준다.

주요 기능으로는 VR 장치의 컨트롤러의 위치와 방향을 연동 가능하게 해주며, 오브젝트와 컨트롤러의 직접적인 상호 작용, 레이저 광선을 이용한 간접적인 상호 작용, Unity UI와의 상호작용 등을 component만으로도 사용할 수 있게 해준다.




[ 그림 39 ] XR Origin의 Component

### 3.3.1.3. XR Interaction Manager

XR Interaction Manager 은 Interactor(상호작용자)와 Interactable(상호작용 대상) 간의 상호작용을 가능하게 하는 핵심 component이다.

VR 사용자와의 상호작용을 관리하며, VR 경험을 더욱 매끄럽게 만든다. Scene 내에 하나의 XR Interaction Manager가 반드시 존재해야 합니다. 만약 플레이 전까지 이 component가 존재하지 않으면, 자동으로 빈 게임 오브젝트에 해당 component를 추가하여 생성된다.

#### 3.3.1.4. XR Input Action Manager

XR Input Action Manager는 연결된 Input Action Asset을 활성화하여 사용자가 정의한 입력을 자동으로 관리해 준다.

스크립트를 통해 수동으로 Input Action을 활성화할 수 있지만, XR Input Action Manager를 사용하면 자동으로 액션을 활성화하고 관리할 수 있다.

#### 3.3.1.5. Locomotion

유니티 XR의 Locomotion은 VR 체험자의 이동을 구현하는 데 사용된다. 이 시스템을 통해 사용자는 VR 공간 내에서 자유롭게 이동하고 방향을 전환할 수 있다.

- **Continuous Move Provider (Action based)**

  사용자가 조이스틱을 앞으로 밀 때 플레이어의 Rig가 어느 방향으로 이동해야 하는지를 정의한다.

  기본적으로 카메라 오브젝트를 사용하여 사용자가 마주 보는 방향 또는 사용자가 컨트롤러를 잡은 방향으로 전진하도록 설정할 수 있다. 이동은 왼쪽 컨트롤러의 조이스틱을 통해 이루어지며, XRI LeftHand Locomotion/Move(Input Action Reference)를 추가하여 설정했다.

- **Continuous Turn Provider**

  조이스틱을 오른쪽으로 기울일 때 VR 플레이어가 매끄럽게 회전할 수 있도록 합니다. 오른쪽 컨트롤러의 조이스틱을 통해 회전하며, XRI RightHand Locomotion/Turn(Input Action Reference)을 추가하여 설정합니다.

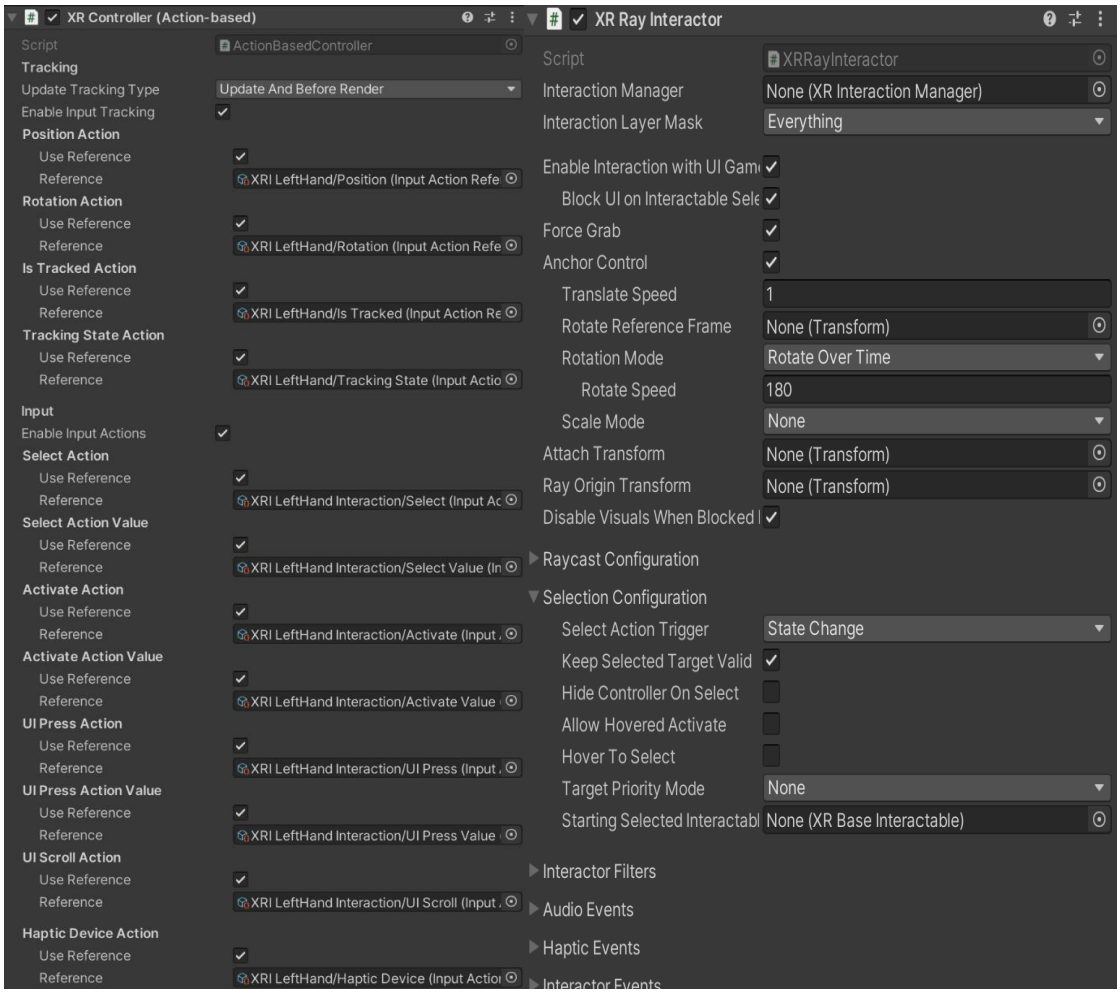
#### 3.3.1.6. Capsule Collider

VR 체험자가 물체와 충돌하는 상황을 감지하기 위해 추가된다. 충돌 감지를 통해 사용자와 VR 환경 간의 상호작용을 보다 현실감 있게 구현할 수 있다.

#### 3.3.1.7. Rigidbody

VR 체험자에게 물리 법칙이 적용되도록 하는 component이다. 사용자가 물리적인 상호

작용을 자연스럽게 느낄 수 있도록 중력, 충돌, 마찰 등의 물리적 요소를 VR 환경에 반영한다.




[ 그림 40 ] Controller의 Component

### 3.3.1.8. XR Controller

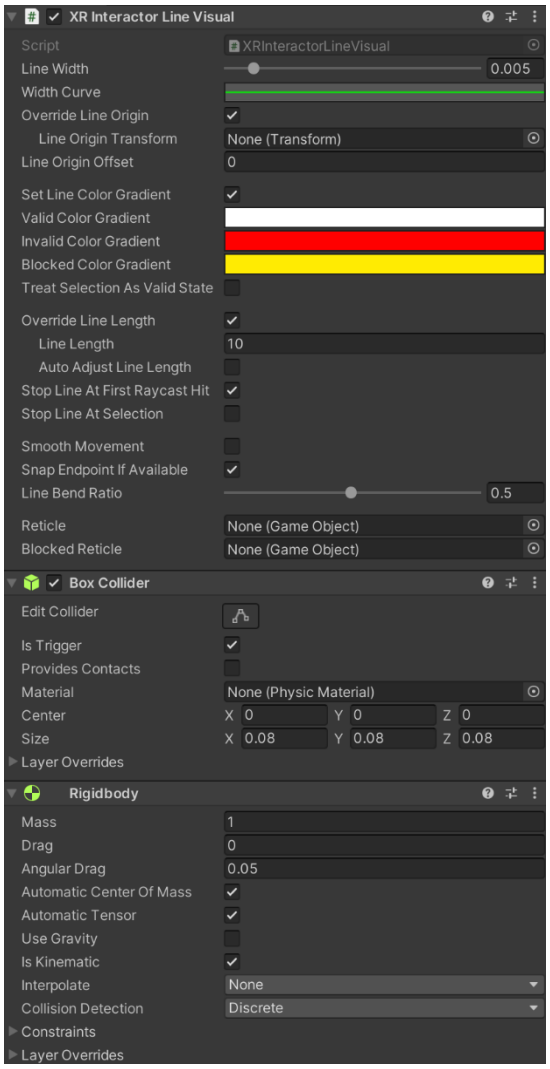
XR Controller는 유니티의 XR Interaction Toolkit의 기본으로 포함된 component로, VR 환경에서 컨트롤러의 동작을 관리하는 역할을 한다.

컨트롤러의 포즈 동기화(pose sync), 그립(Grip) 버튼, 트리거(Trigger) 버튼 등의 액션을 연결할 수 있다. 기본적으로 XRI Default Input Action Asset이 연결되어 있어 별도의 추가 작업 없이 기본적인 상호작용을 구현할 수 있다. 왼쪽, 오른쪽 컨트롤러에 각각 맞는 reference를 추가하여 사용자의 동작을 인식하고 적용한다.

### 3.3.1.9. XR Ray Interactor

XR Ray Interactor는 컨트롤러에서 나오는 광선(Ray)을 상호작용 가능한 물체와 연결하여, 해당 물체와의 상호작용을 가능하게 한다.

광선이 상호작용 가능한 물체에 닿으면, 해당 물체와 상호작용을 할 수 있도록 제어한다. 이에 따라 VR 사용자들은 물리적으로 접근하지 않고도 컨트롤러를 통해 먼 거리에서 물체와 상호작용을 할 수 있다.



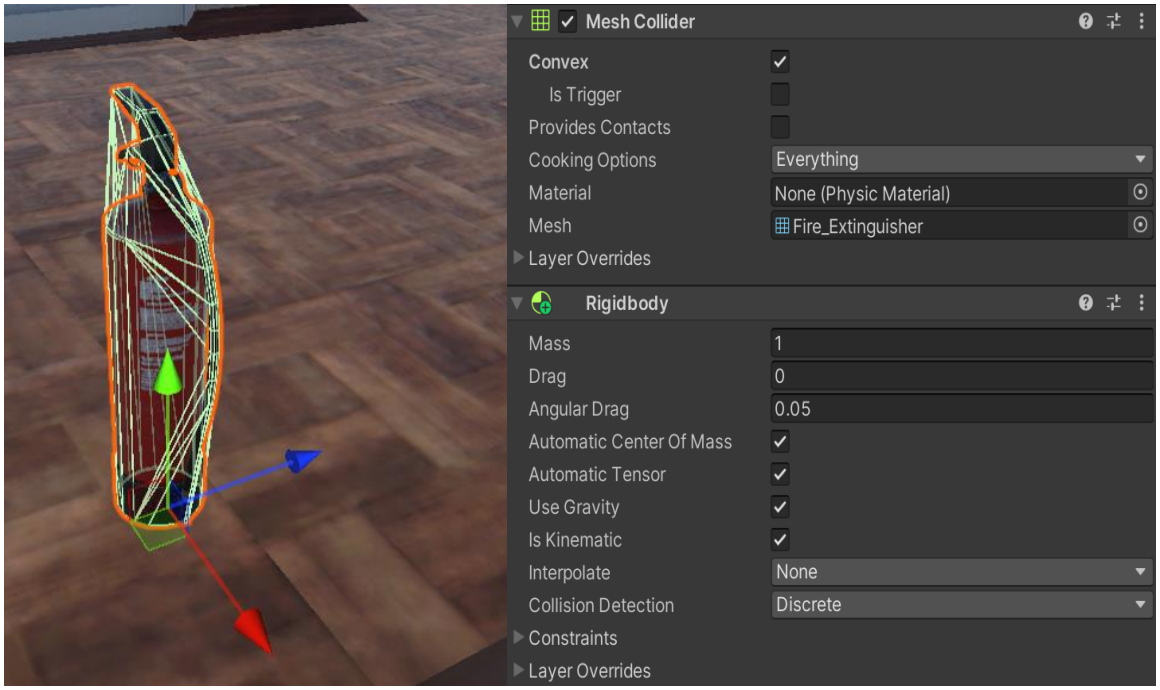

[ 그림 41 ] Controller의 Component

#### 3.3.1.10. XR Interactor Line Visual

XR Interactor Line Visual은 컨트롤러에서 나오는 광선을 조작하고 시각적으로 표현하는 component이다.

사용자가 상호작용을 할 수 있는 물체와 닿았을 때, 광선의 색상을 변경하여 피드백을 제공한다. 예를 들어, 상호작용 불가 상태일 때는 빨간색으로 표시되고, 상호작용 가능한 상태가 되면 광선이 흰색으로 변하게 설정할 수 있다. 이를 통해 체험자는 VR 환경에서 상호작용 가능한 물체를 쉽게 구분할 수 있다.

### 3.3.2. XR Interaction



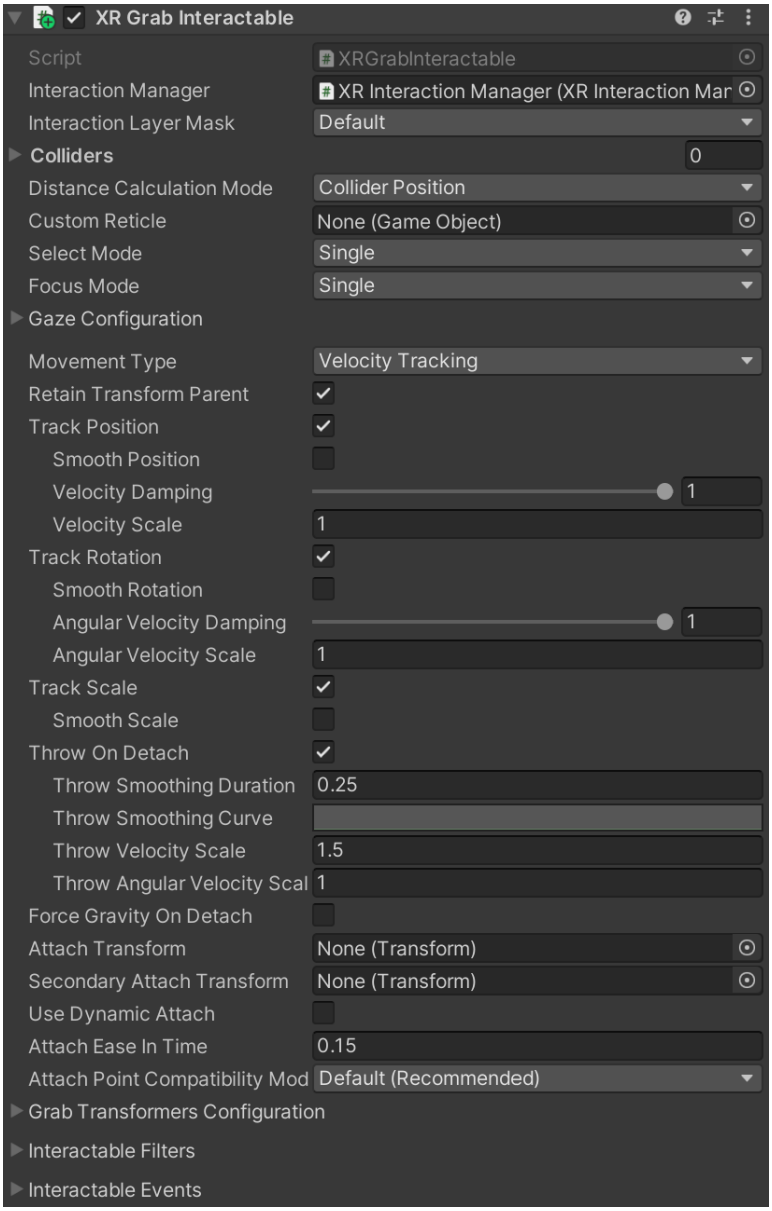

[ 그림 42 ] 소화기의 Component

#### 3.3.2.1. Mesh Collider

Mesh Collider는 VR 체험자가 상호작용을 할 물체에 충돌 감지 기능을 추가하는 component이다. VR 환경에서 체험자와 상호작용을 하고자 하는 모든 물체에 Collider를 추가하여 충돌을 감지할 수 있습니다.

소화기처럼 복잡한 형태의 물체는 Mesh collider를 추가하여 충돌 처리를 세밀하게 구

현했다. 사용자가 VR 환경 내에서 더욱 자연스럽고 현실적인 상호작용을 경험할 수 있다.



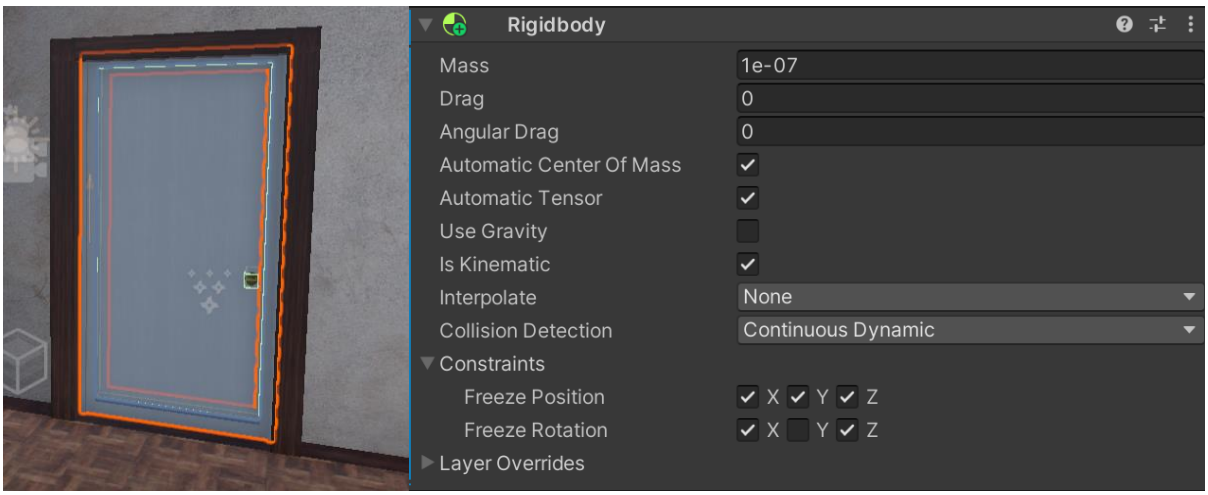

[ 그림 43 ] 소화기의 Component

### 3.3.2.2. XR Grab Interactable

XR Grab Interactable은 VR 환경에서 물체를 잡고(Grab), 던질 수 있는 기본적인 상호작용을 가능하게 해주는 component이다. 이 component는 물체에 Rigidbody 컴포넌트를 자동으로 추가하여, 물리적 상호작용을 처리한다. 이를 통해 물체는 중력과 같은 물리 법

칙도 따르게 된다.

XR Interaction Manager에 설정된 컨트롤러의 그립(Grip) 버튼을 이용하여 물체를 잡을 수 있습니다. 사용자가 그립 버튼을 누르면 물체를 잡고, 버튼을 떼면 물체를 던지거나 놓을 수 있다. 이러한 그립 상호작용을 통해 사용자에게 물체를 다루는 행동을 직관적이고 자연스럽게 느낄 수 있게 한다.

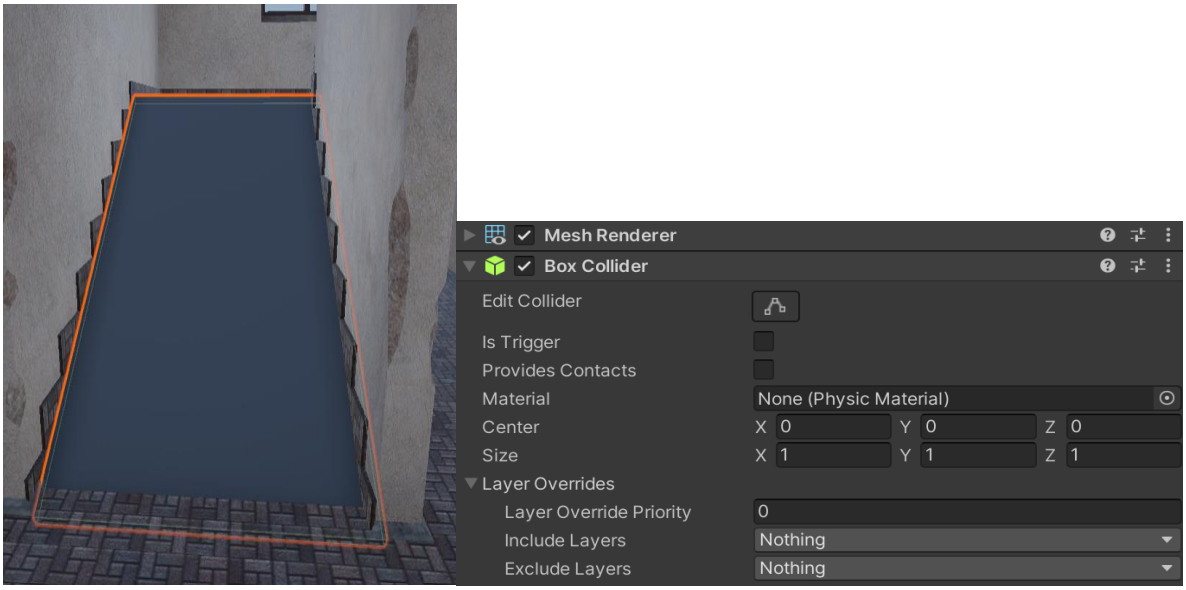



[ 그림 44 ] 문의 Component

### 3.3.2.3. 회전하는 물체의 Rigidbody

회전하는 물체(예: 문)와 같은 상호작용이 필요한 물체는 Collider를 통한 충돌 감지 기능 외에도 물리적 회전을 처리하기 위한 Rigidbody component 설정이 필요하다.

특히 문은 손잡이가 있는 쪽 반대 방향으로 y축을 기준으로 회전이 필요하므로, y축을 제외한 모든 축의 회전과 이동을 고정한다. 이를 통해 문이 예상하지 않은 방향으로 움직이지 않도록 제어할 수 있다. 이를 통해 체험자는 문을 현실 세계에서처럼 부드럽게 회전시키며 여닫을 수 있다.




[ 그림 45 ] 계단의 Component

#### 3.3.2.4. 경사면의 Collider

계단과 같은 복잡한 구조물은 정확한 충돌 감지를 위해 여러 개의 collider를 사용하는 것이 비효율적이다. 대신, 간단한 형태의 경사면을 Box Collider를 사용하여 구현함으로써 VR 체험자가 경사면을 오르고 상호작용을 할 수 있도록 한다.

이에 따라 체험자가 경사면을 자연스럽게 오르며 상호작용을 할 수 있다. 경사면은 복잡한 구조물 위에서도 물리적인 충돌을 처리할 수 있어 효율적인 충돌 감지와 함께 VR 환경의 몰입감을 유지할 수 있다.

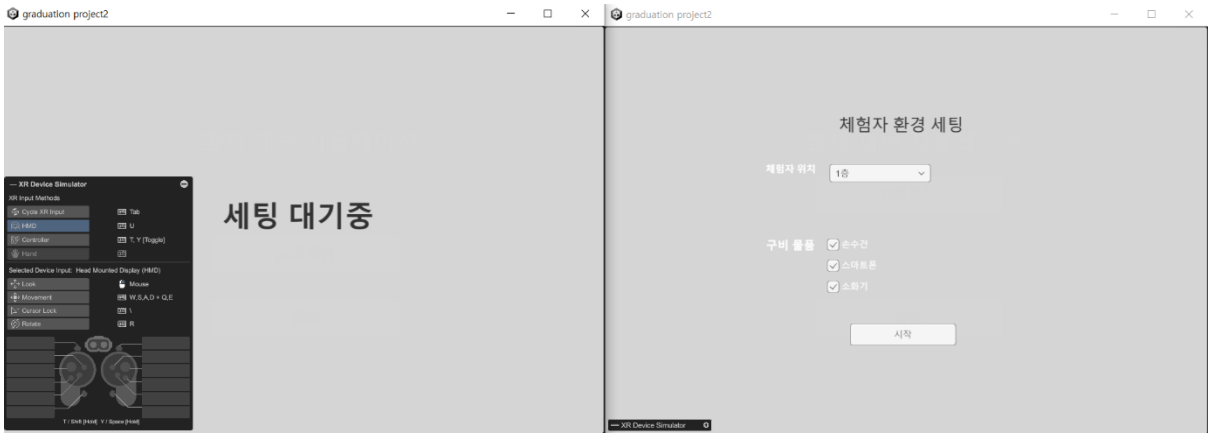
## 3.4. 예상 시나리오

### 3.4.1. 예상 시나리오 1 (간단한 탈출 상황)

감독관이 세팅 화면에서 체험자의 위치를 1층으로 설정하고, 구비 물품으로 스마트폰을 제공한 후 관리 화면에서 2~3층에 불을 스폰 시킨다면 체험자는 즉시 탈출이 가능하고 위험하지 않은 간단한 상황이고, 체험자는 스마트폰을 이용하여 119에 전화하는 시늉을 하고 긴급하게 탈출해야 하는 시나리오로, 만약 체험자가 상황을 제대로 파악하지 못해 긴급하게 탈출하지 않거나 119에 신고하지 않는다면 음성채팅을 통해 피드백을 제공할 수 있다.

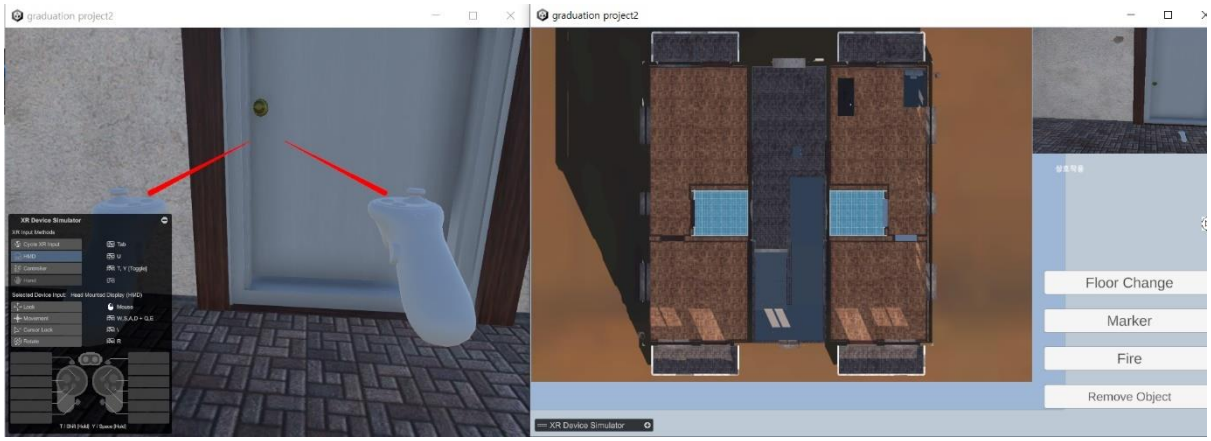
### 3.4.2. 예상 시나리오 2 (탈출이 힘든 상황)

감독관이 세팅 화면에서 체험자의 위치를 3층으로 설정하고, 구비 물품으로 소화기, 손수건을 제공한 후 관리 화면에서 1층 입구에 불을 스폰 시킨다면 체험자는 탈출이 힘든 상황이므로, 체험자는 손수건으로 입을 막는 시늉을 하고 상황을 파악해야 할 것이며, 소화기를 이용하여 불을 끄거나, 스마트폰이 제공되지 않았으므로 옥상으로 대피하여 구조요청을 하는 등 다양한 대처를 해야 하는 시나리오로, 감독관은 마커를 이용하여 체험자가 소화기의 위치를 모를 때 표시하거나 가야 할 길의 위치를 알려줄 수 있고, 체험자가 잘못된 행동을 했을 경우 음성채팅을 통해 즉시 피드백을 제공할 수 있다.



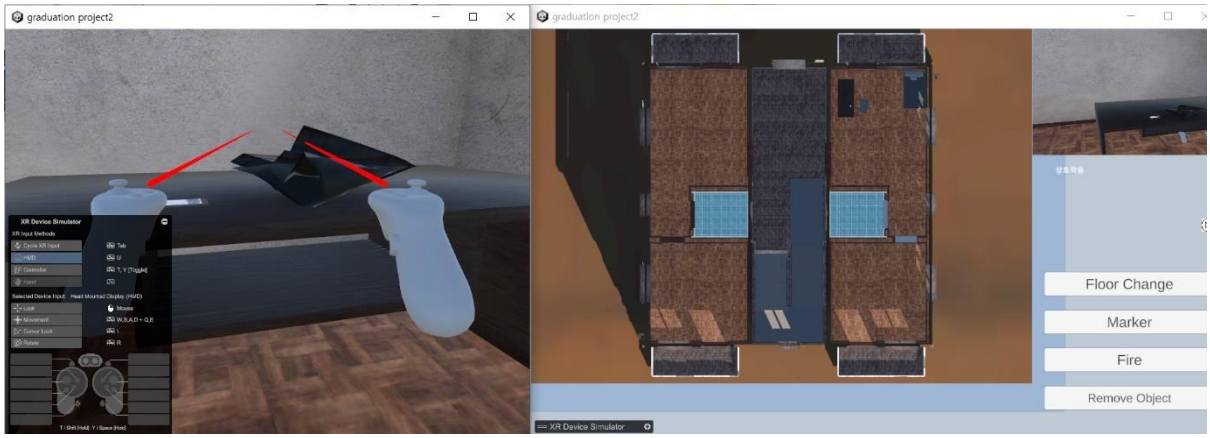

[ 그림 46 ] 체험자화면(좌), 감독관화면(우)

체험자와 감독관이 각각 실행 후 본인의 역할에 맞는 버튼을 누르면 [그림 46]과 같다.



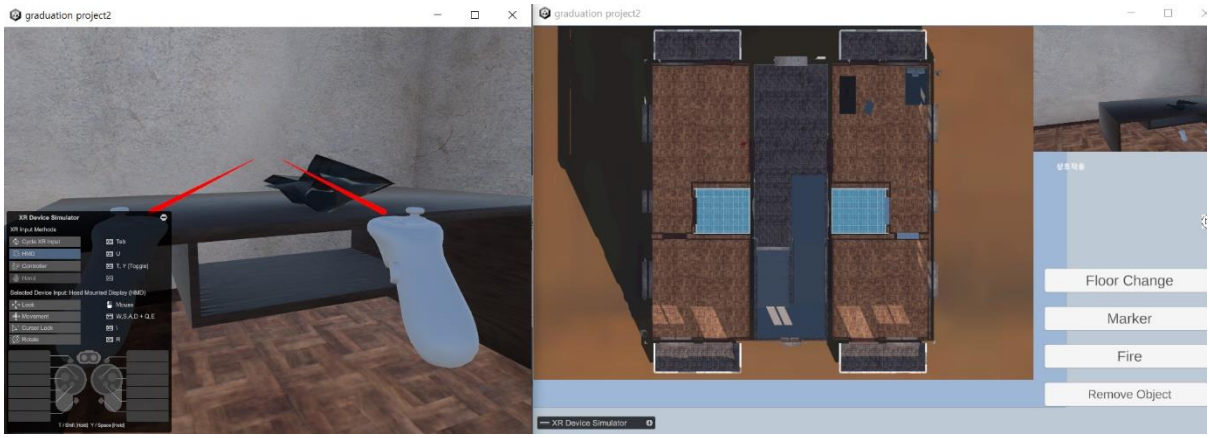

[ 그림 47 ] 감독관이 시작 버튼을 누른 후

감독관이 시작 버튼을 누름과 동시에 체험자는 vr화면으로 진입하고 감독관은 관리 화면으로 들어간다. 미니맵에서 남색 직사각형이 체험자를 따라가게 만들어 위치를 명확하게 알 수 있다.



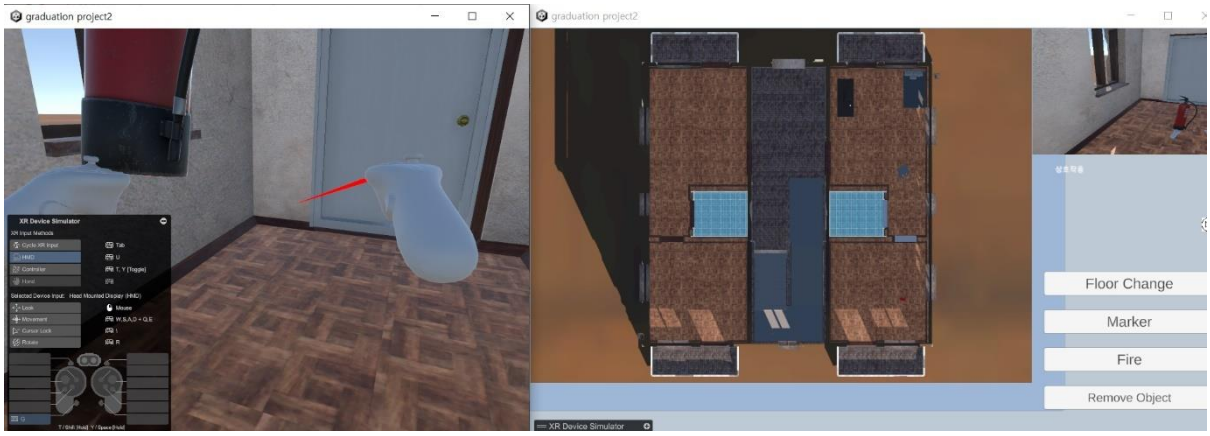

[ 그림 48 ] 손수건과 스마트폰 모두 존재

[그림 46]의 세팅 화면에서 감독관이 모든 물건을 모두 체크하였으므로 [그림 48]과 같이 체험자는 모든 물건을 이용할 수 있다.



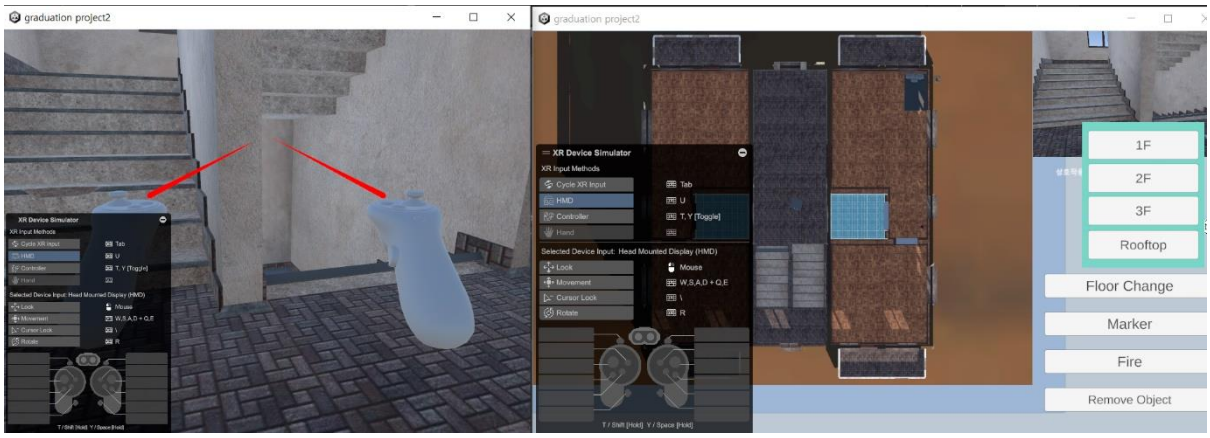

[ 그림 49 ] 스마트폰 체크를 해제하여 사라짐

만약 감독관이 세팅 화면에서 예시로 스마트폰을 해제할 경우 [그림48]과 달리 [그림49]와 같이 체험자에게는 스마트폰이 보이지 않고 이용할 수 없게 된다.



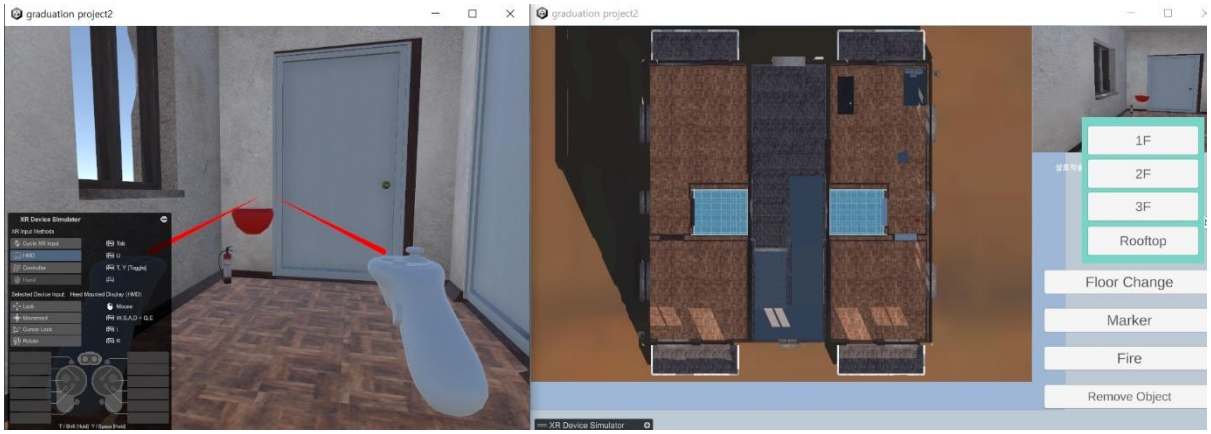

[ 그림 50 ] 체험자의 상호작용

체험자는 [그림50]과 같이 구비된 물건과 상호작용할 수 있다.



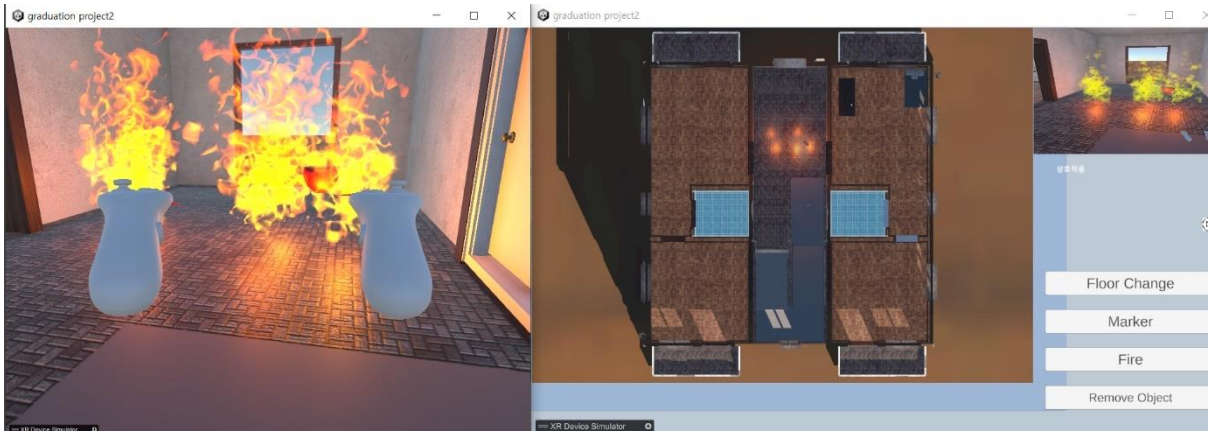

[ 그림 51 ] 체험자 2층세팅, floor change 2층 버튼 사용

또한 만약 감독관이 세팅 화면에서 예시로 체험자의 층수를 2층으로 바꿨다면 체험자는 2층에서 시작되고 Floor change 기능을 사용하여 2층을 누른다면 [그림 51]과 같이 감독관은 2층에 있는 체험자를 확인할 수 있다.




[ 그림 52 ] 마커를 통한 의사소통

마커 버튼을 미니맵에 원하는 곳에 누르면 [그림 52]와 같이 체험자는 마커를 확인할 수 있다.




[ 그림 53] 감독관의 불 생성

마찬가지로 Fire버튼을 미니맵에 누르면 [그림 53]과 같이 생성된 불을 확인할 수 있다.

## 4. 구현된 기능의 동작 및 문제점

본 연구에서 개발한 VR 기반 재난 시뮬레이션 시스템은 여러 주요 기능들이 제대로 동작하여 체험자와 감독관 간의 원활한 상호작용을 가능하게 했다. 다만 VR 기기 연동 문제와 일부 기능에서 발생한 충돌 판정 문제, 그리고 몰입감의 부족이 개선이 필요한 부분으로 확인되었다.

## 4.1. 정상 동작하는 부분

### 1. 네트워크 연결 및 상호작용

VR 체험자와 감독관 간의 네트워크 연결은 Unity Netcode for GameObjects를 통해 원활하게 이루어졌다. 체험자는 클라이언트로, 감독관은 호스트로 접속하여 실시간으로 데이터를 주고받으며 상호작용을 할 수 있었다. 특히, 감독관은 체험자의 진행 상황을 실시간으로 모니터링하고, 미니맵을 통해 건물의 단면도를 보며 체험자의 위치를 파악하고, 음성 통신을 통해 실시간 피드백을 제공하는 기능이 원활하게 동작했다.

### 2. 미니맵과 마커 시스템

감독관이 사용하는 미니맵과 마커 스폰 시스템은 예상대로 동작하여, 체험자가 재난 상황에서 길을 잃지 않고 특정 위치로 안내될 수 있었다. 감독관이 특정 위치를 클릭하면 그 위치에 마커가 생성되어 체험자 화면에 즉시 표시되며, 이전에 생성된 마커는 자동으로 사라지고 새로운 마커가 생성되는 기능이 정상적으로 구현되었다. 이 기능을 통해 체험자는 시뮬레이션 중 길 안내를 받으며 더 빠르게 행동할 수 있었다.

### 3. 화재 스폰 및 제거 시스템

감독관이 재난 상황을 시뮬레이션하기 위해 특정 위치에 불 오브젝트를 스폰하는 기능도 정상적으로 작동했다. 불은 최대 5개까지 스폰할 수 있으며, 추가로 화재를 생성하면 가장 먼저 생성된 불이 사라지는 로직이 제대로 구현되었다. 이 기능을 통해 감독관은 실시간으로 시뮬레이션 난이도를 조절하고, 체험자가 변화하는 재난 상황에 어떻게 대응하는지를 평가할 수 있었다.

### 4. 음성 채팅 시스템

Vivox를 통한 음성 채팅 기능도 안정적으로 작동했다. 체험자와 감독관은 음성 채팅을 통해 실시간으로 의사소통 하며 시뮬레이션 진행 중 필요한 피드백을 주고받을 수 있었다. 음성 채팅을 통한 의사소통은 감독관이 시뮬레이션 중 체험자의 행동을 즉각적으로 수정하거나 지시할 수 있어 교육적 효과를 높이는 데 기여했다.

**5. UI 시스템**

VR 체험자의 대기 화면과 세팅 UI, 그리고 시뮬레이션이 시작되기 전 감독관이 위치와 상황을 설정할 수 있는 세팅 UI가 원활하게 동작했다. 이로 인해 체험자는 대기 상태에서 연결을 기다릴 수 있었고, 감독관은 체험자가 시작할 위치와 필요한 도구들을 설정하여 시뮬레이션을 맞춤형으로 조정할 수 있었다.

**6. VR 체험자의 기본 조작 및 상호작용**

Oculus를 통한 VR 체험자의 시선은 Unity와 연동되어, 체험자의 머리 움직임에 따라 시점이 자연스럽게 동작했다. 연결된 Oculus 컨트롤러를 사용하여 체험자는 가상환경 내에서 자유롭게 이동할 수 있으며, 기본적인 상호작용으로 물체를 잡고 던지는 기능도 구현되었다.

## 4.2. 문제점

**1. VR 기기 연동 문제**

시스템을 Oculus Quest 2에 빌드했지만 실행되지 않는 문제가 발생했다. 빌드 과정은 오류 없이 완료되었으나, 기기에서 애플리케이션이 정상적으로 실행되지 않았다. 원인은 정확히 파악되지 않았지만, 버전 호환성 문제 또는 추가적인 기기 설정이 누락되었을 가능성이 있다. Oculus Integration과 Unity 버전 간의 호환성 문제를 해결하는 것이 필요하다.

**2. 충돌 판정 문제**

XR Origin과 Capsule Collider를 이용한 체험자의 충돌 판정에서 오류가 발생했다. 체험자가 벽을 통과하거나 계단을 올라가지 못하는 문제가 있었고, 충돌 처리가 예상대로 이루어지지 않았다. 이는 몰입감을 저해하고, 체험자가 시뮬레이션을 실제 상황처럼 경험하는 데 어려움을 주는 큰 문제로 작용했다.

## 4.3. 개선될 점

1. **VR 기기 연동 안정화**

   Oculus Quest 2에서 실행되지 않는 문제를 해결하기 위해 Unity와 Oculus Integration 간의 버전 호환성을 확인하고, 필요한 추가 설정을 점검해야 한다. 기기와의 안정적인 연동을 통해 체험자가 재난 시나리오를 제대로 체험할 수 있도록 해야 한다.

2. **충돌 판정 문제 해결**

   체험자의 Capsule Collider와 XR Origin의 물리 엔진 문제를 해결할 필요가 있다. 충돌 판정이 제대로 이루어지지 않는 이유를 파악하고, 물리 엔진 및 Collider 설정을 재검토하여 체험자가 물체와 자연스럽게 상호작용을 할 수 있도록 수정해야 한다.

3. **세부 상호작용 추가**

   체험자가 몰입감을 더 느낄 수 있도록, 재난 상황에서 실제로 필요한 스마트폰 조작, 소화기 사용, 문 열기 등의 상호작용을 추가해야 한다. 이러한 기능들은 체험자가 보다 현실감 있게 훈련을 진행할 수 있게 하며, 시뮬레이션의 실용성을 높인다.

4. **메뉴 시스템 개선**

   체험자가 사용할 수 있는 인벤토리나 시뮬레이션 설정 메뉴를 추가하여 시뮬레이션을 더 세부적으로 관리할 수 있는 시스템이 필요하다. 이를 통해 체험자는 재난 상황에서 다양한 도구를 사용할 수 있으며, 보다 복합적인 시나리오에 대응할 수 있다.

이와 같이, 잘 구현된 부분과 문제점, 개선될 점을 종합적으로 분석하면 시스템의 완성도를 더욱 높일 수 있을 것이다.

# 5. 결론 및 향후 과제 방향

## 5.1. 결론

본 과제에서는 VR 기반 재난 상황 교육/예방 시뮬레이션 시스템을 개발하였다. 기존의 단순한 영상이나 이론 학습을 통한 재난 교육 방식에 비해, VR 기반의 시뮬레이션이 체험자에게 실제와 유사한 재난 상황을 제공하여 몰입감을 높이고 교육 효과가 향상했다. 이 과정에서 체험자의 몰입감을 높이기 위한 다양한 상호작용 방식을 연구하였고, 이를 통해 재난 상황에서 실전 감각을 키우고 교육 효과를 극대화할 수 있었다.

특히, 본 시스템은 실시간으로 감독관과 체험자가 상호 피드백을 주고받을 수 있는 기능을 추가하여, 체험자가 각기 다른 상황에 맞춤형으로 대응할 수 있도록 설계되었다. 이를 통해 교육의 정확성과 실효성을 크게 향상할 수 있었으며, 체험자는 가상 환경에서의 직접적인 상호작용을 통해 실전 대응 능력을 강화할 수 있었다. 또한, 상황별 몰입감을 고려하여 시나리오 설계와 상호작용 피드백을 최적화함으로써, 교육의 효과를 더욱 높이고자 했다.

## 5.2. 산학 협력 멘토링

초기에 설정했던 과제 주제는 "RGBD 카메라를 이용한 3D User Interaction 기법 연구"였으나, 산학협력 멘토링을 통해 새로운 측면에서 주제를 재구성할 수 있었다. 멘토링을 통해 RGBD 카메라만으로는 현실 세계에서의 체험자와의 물리적 상호작용을 구현하는데 한계가 있으며, 이를 통해 몰입감을 극대화하기 어려울 수 있다는 조언을 받았다. 이에 따라, 콘텐츠 몰입감을 효과적으로 증대할 수 있는 새로운 시스템을 모색했다.

산학협력체에서 실제로 적용하고 있는 실시간 솔루션 사례들을 바탕으로 "VR 기반 재난 상황 교육 시스템"과 실시간 네트워크 솔루션을 결합하는 아이디어를 고안하게 되었다. 이를 통해 현실 세계의 사용자가 가상 환경에서 더욱 몰입감 있게 상호작용을 할 수 있는 시스템을 설계할 수 있었고, 이러한 시스템은 재난 상황을 더욱 현실적으로 체험하며 학습하는 데 중요한 역할을 할 것으로 기대된다.

## 5.3. 향후 과제 방향

향후에 VR 체험자가 이용할 수 있는 물체와 다양한 상호작용을 추가한다면, 더욱 다채

로운 체험이 가능해져 교육 효과뿐만 아니라 콘텐츠의 몰입감도 한층 더 높일 수 있을 것으로 예상된다. 또한, 화재 상황뿐만 아니라 지진, 지하철 사고 등 다양한 재난 상황을 시뮬레이션에 추가한다면, 더 많은 사람들이 본 시스템을 활용할 수 있을 것으로 기대된다. 이러한 개선을 통해 보다 포괄적이고 실질적인 재난 대응 교육 시스템이 될 수 있을 것이다.

# 6. 개발 일정 및 역할 분담

## 6.1. 개발일정

| 주요일정 | 6월 | | | | 7월 | | | | 8월 | | | | 9월 | | | | 10월 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Unity3D 사용법 및 C# 숙지 | | | | | | | | | | | | | | | | | | | | |
| 시나리오 구성 | | | | | | | | | | | | | | | | | | | | |
| 3D 가상환경 구현 | | | | | | | | | | | | | | | | | | | | |
| UI 구현 | | | | | | | | | | | | | | | | | | | | |
| 중간 보고서 작성 | | | | | | | | | | | | | | | | | | | | |
| 네트워크 구현 | | | | | | | | | | | | | | | | | | | | |
| 3D Interaction 구현 및 수정 | | | | | | | | | | | | | | | | | | | | |
| 오류 확인 및 최종 테스트 | | | | | | | | | | | | | | | | | | | | |
| 최종보고서 작성 및 발표 준비 | | | | | | | | | | | | | | | | | | | | |

### 6.2. 역할 분담

| 이름 | 역할 |
|---|---|
| 이종민 | - 기기 간 네트워크 연결 및 세팅(Netcode)<br>- 1:1 의사소통 기법 구현(미니맵, 마커)<br>- UI 디자인 및 최적화<br>- 음성 채팅 구현(Vivox) |
| 노윤정 | - VR 작동 환경 구현<br>- 가상환경 속 상호작용 구현<br>- 가상환경 수정 및 보완(벽, 바닥, 계단 등) |
| 장승우 | - 시나리오 구성<br>- ui구성 및 동작<br>- 네트워크 보조 (체험자 따라가는 카메라)<br>- 환경세팅(체험자 위치이동, 물건 배치) |
| 공통 | - Unity 3D 기초 지식 공부<br>- 정보 수집 및 논문 분석<br>- 보고서 작성 및 발표 준비 |

## 7. 참고 문헌


[1] H. Dong. (2023, Apr 11). "[Unity 3D/VR] XR Interaction Toolkit 소개 및 프로젝트 세팅." [Online]. Available: https://howudong.tistory.com/258 (accessed 2024, Oct. 15).

[2] Y. Kim. (2023, Apr 10). "유니티 멀티플레이의 기반인 Netcode 라이브러리 정리." [Online]. Available: https://kimyir.tistory.com/42 (accessed 2024, Oct. 15).

[3] Unity. (2022). "Unity User Manual 2022.1_Vivox." [Online]. Available: https://docs.unity3d.com/kr/2022.1/Manual/com.unity.services.vivox.html (accessed 2024, Oct. 15).

[4] Virnect. (2024, Oct 15). "Store Board - Semiconductor." [Online]. Available: https://store.virnect.com/board/view?id=semiconductor&seq=660 (accessed 2024, Oct. 15).

[5] Unity Technologies. (2023). "Unity Documentation." [Online]. Available: https://docs.unity3d.com (accessed 2024, Oct. 15).

[6] Oculus. (2024). "Oculus Developer Documentation." [Online]. Available: https://developer.oculus.com (accessed 2024, Oct. 15).

[7] K. Fire Safety Institute. (2020, Dec). "VR/XR 기반 재난 안전 훈련기술에 관한 고찰." [Online]. Available: https://www.kfsi.or.kr/contents/webzine/202012/sub02-02.html (accessed 2024, Oct. 15).

[8] ICT XR Center. (n.d.). "부산 어린이 VR 재난 안전 체험 교육장." [Online]. Available: http://ictxr.or.kr/sub/?mcode=0404010000 (accessed 2024, Oct. 15).

[9] SG Safety. (n.d.). "중대재해예방 VR 교육." [Online]. Available: https://www.vrsg.co.kr/VR%EC%95%88%EC%A0%84%EA%B5%90%EC%9C%A1-%EC%86%8C%EA%B0%9C (accessed 2024, Oct. 15).

[10] Korea Occupational Safety and Health Agency (KOSHA). (n.d.). "VR 전용관." [Online]. Available: https://360vr.kosha.or.kr/main (accessed 2024, Oct. 15).

[11] Samwoo Immersion. (n.d.). "몰입체험 VR 컨텐츠 (청소년 안전 체험팩 - 재난안전)." [Online]. Available: https://www.samwooim.com/IMXR_all/?q=YToxOntzOjEyOiJrZXl3b3JkX3R5cGUiO3M6MzoiYWxsIjt9&bmode=view&idx=15450749&t=board (accessed 2024, Oct. 15).

[12] Digital Daily. (2021, May 31). "'위험한 현장 투입전에 VR로 미리 체험, 확실히 도움된다'…건설·중공업계 '교육' 혁신." [Online]. Available: https://m.ddaily.co.kr/page/view/20210531110236113343 (accessed 2024, Oct. 15).

[13] Mline Studio. (n.d.). "한국산업안전보건공단 VR 가상현실 안전보건교육." [Online]. Available: https://m-line.tv/portfolio_page/2023g005/ (accessed 2024, Oct. 15).